MEMORY FORENSICS

A Comprehensive Technical Guide

CHRISTMAS EDITION

HADESS

WWW.HADESS.IO

INTRODUCTION

In the ever-evolving landscape of cybersecurity, memory forensics has emerged as a pivotal technique in digital investigations. Unlike traditional disk forensics, which focuses on analyzing static data, memory forensics dives deep into the volatile memory (RAM) of a system. This approach is essential for uncovering evidence of malicious activity, such as active malware, encryption keys, and transient data, that resides exclusively in memory and disappears upon power-off. As cyberattacks grow more sophisticated, memory forensics has become an indispensable tool for incident responders and forensic investigators alike.

At its core, memory forensics enables the extraction and analysis of system states during live operations. This is critical for detecting advanced threats such as rootkits, process injection, and fileless malware, which are specifically designed to avoid detection on storage media. By capturing a snapshot of a system's memory, forensic analysts can reconstruct the events leading up to a breach and identify suspicious activities that might otherwise leave no trace. Tools like Volatility, Rekall, and modern commercial solutions have streamlined this process, offering investigators powerful capabilities for examining volatile data across various operating systems.

This comprehensive guide delves into the technical aspects of memory forensics, offering insights into its methodologies, tools, and real-world applications. Whether you are an incident responder, a malware analyst, or a digital forensics professional, this article provides a detailed roadmap for leveraging memory forensics in combating modern cyber threats. From understanding memory structures to employing cutting-edge tools and techniques, this guide aims to equip readers with the knowledge required to excel in the field of volatile memory analysis.

To be the vanguard of cybersecurity, Hadess envisions a world where digital assets are safeguarded from malicious actors. We strive to create a secure digital ecosystem, where businesses and individuals can thrive with confidence, knowing that their data is protected. Through relentless innovation and unwavering dedication, we aim to establish Hadess as a symbol of trust, resilience, and retribution in the fight against cyber threats.

DOCUMENT INFO

HADESS

To be the vanguard of cybersecurity, Hadess envisions a world where digital assets are safeguarded from malicious actors. We strive to create a secure digital ecosystem, where businesses and individuals can thrive with confidence, knowing that their data is protected. Through relentless innovation and unwavering dedication, we aim to establish Hadess as a symbol of trust, resilience, and retribution in the fight against cyber threats.

At Hadess, our mission is twofold: to unleash the power of white hat hacking in punishing black hat hackers and to fortify the digital defenses of our clients. We are committed to employing our elite team of expert cybersecurity professionals to identify, neutralize, and bring to justice those who seek to exploit vulnerabilities. Simultaneously, we provide comprehensive solutions and services to protect our client's digital assets, ensuring their resilience against cyber attacks. With an unwavering focus on integrity, innovation, and client satisfaction, we strive to be the guardian of trust and security in the digital realm.

Security Researcher
Diyar Saadi

KNOWLEDGE IS POWER

TABLE OF CONTENT

- Memory Structure
- Volatility Essentials
- Core Investigation Plugins
- Custom Plugin Development
- Practical Memory Analysis Workflows
- Investigation of Running Processes
- Advanced Memory Forensics Techniques
- Enhanced Cellebrite Memory Acquisition
- Enhanced FTK Imager Memory Acquisition
- Enhanced Volatile Memory Acquisition
- Container and Cloud Memory Analysis
- Memory Forensics Automation

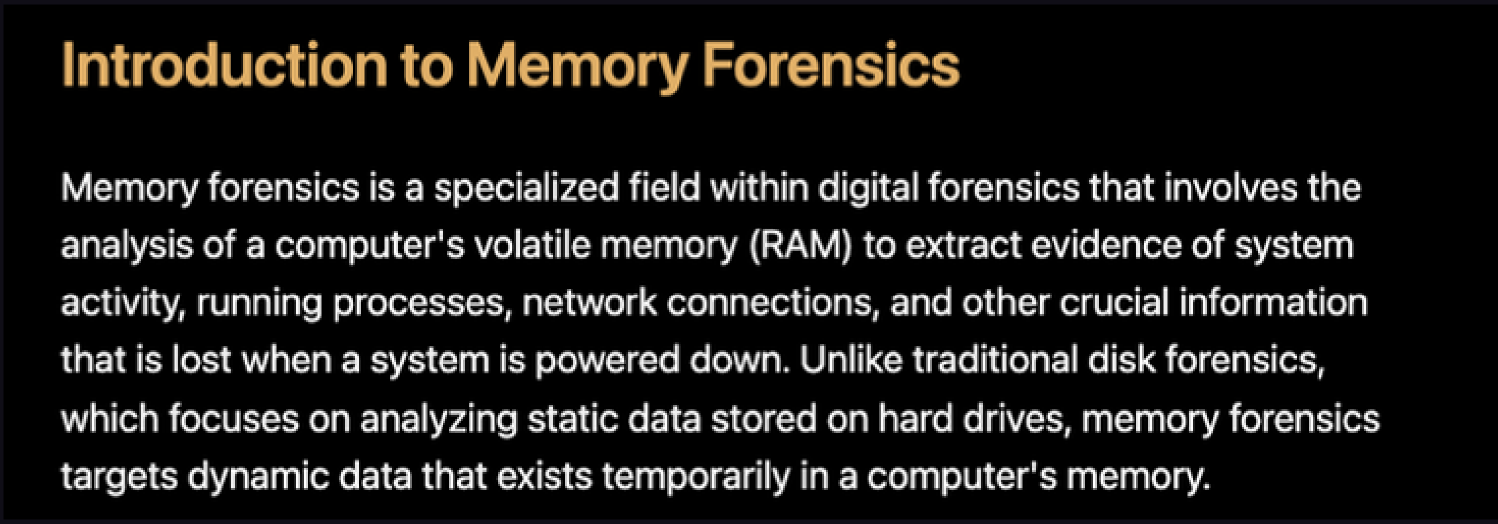
EXECUTIVE SUMMARY

Memory forensics has become a critical component in modern cybersecurity investigations, offering unparalleled insights into system activity and volatile data. Unlike traditional disk forensics, memory forensics focuses on capturing and analyzing the contents of a system's RAM to uncover evidence of active threats, such as malware, rootkits, and encryption keys. This process is instrumental in identifying sophisticated attack vectors, including fileless malware and process injections, which often evade traditional detection mechanisms. As cyber threats grow more advanced, the demand for memory forensics expertise continues to rise, making it a vital skill for incident responders and digital forensic professionals.

This technical guide explores the key principles, methodologies, and tools involved in memory forensics. From acquiring memory images using tools like FTK Imager and Cellebrite to analyzing volatile data with frameworks like Volatility and Rekall, the article provides a step-by-step roadmap for mastering this specialized domain. Additionally, it highlights practical applications, including incident response, malware analysis, and threat hunting, while addressing the challenges and best practices for effective memory analysis. Whether investigating live incidents or reconstructing post-breach scenarios, memory forensics is an indispensable resource for staying ahead in the fight against cybercrime.

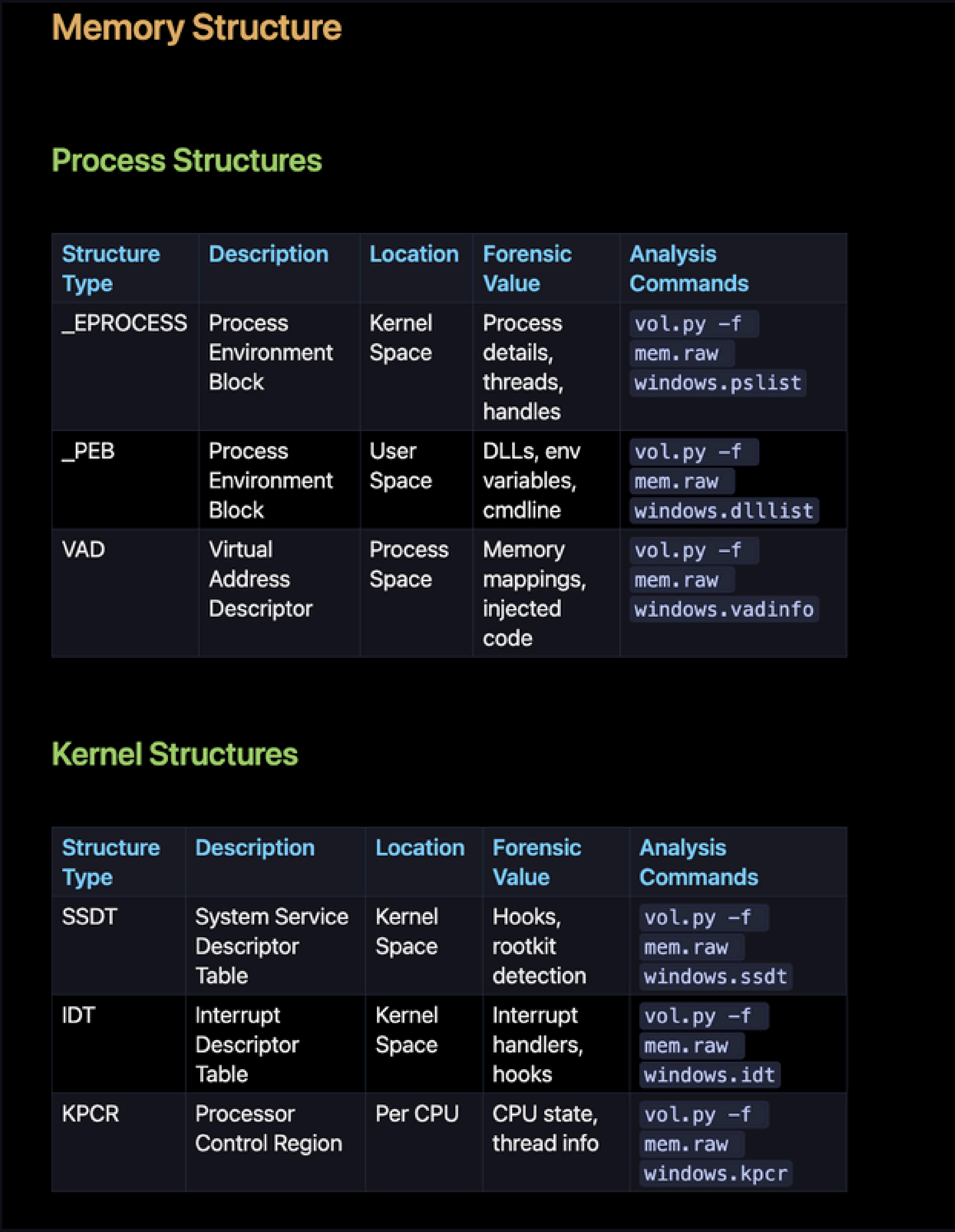
01

ATTACKS



Introduction to Memory Forensics

Memory forensics is a specialized field within digital forensics that involves the analysis of a computer's volatile memory (RAM) to extract evidence of system activity, running processes, network connections, and other crucial information that is lost when a system is powered down. Unlike traditional disk forensics, which focuses on analyzing static data stored on hard drives, memory forensics targets dynamic data that exists temporarily in a computer's memory.



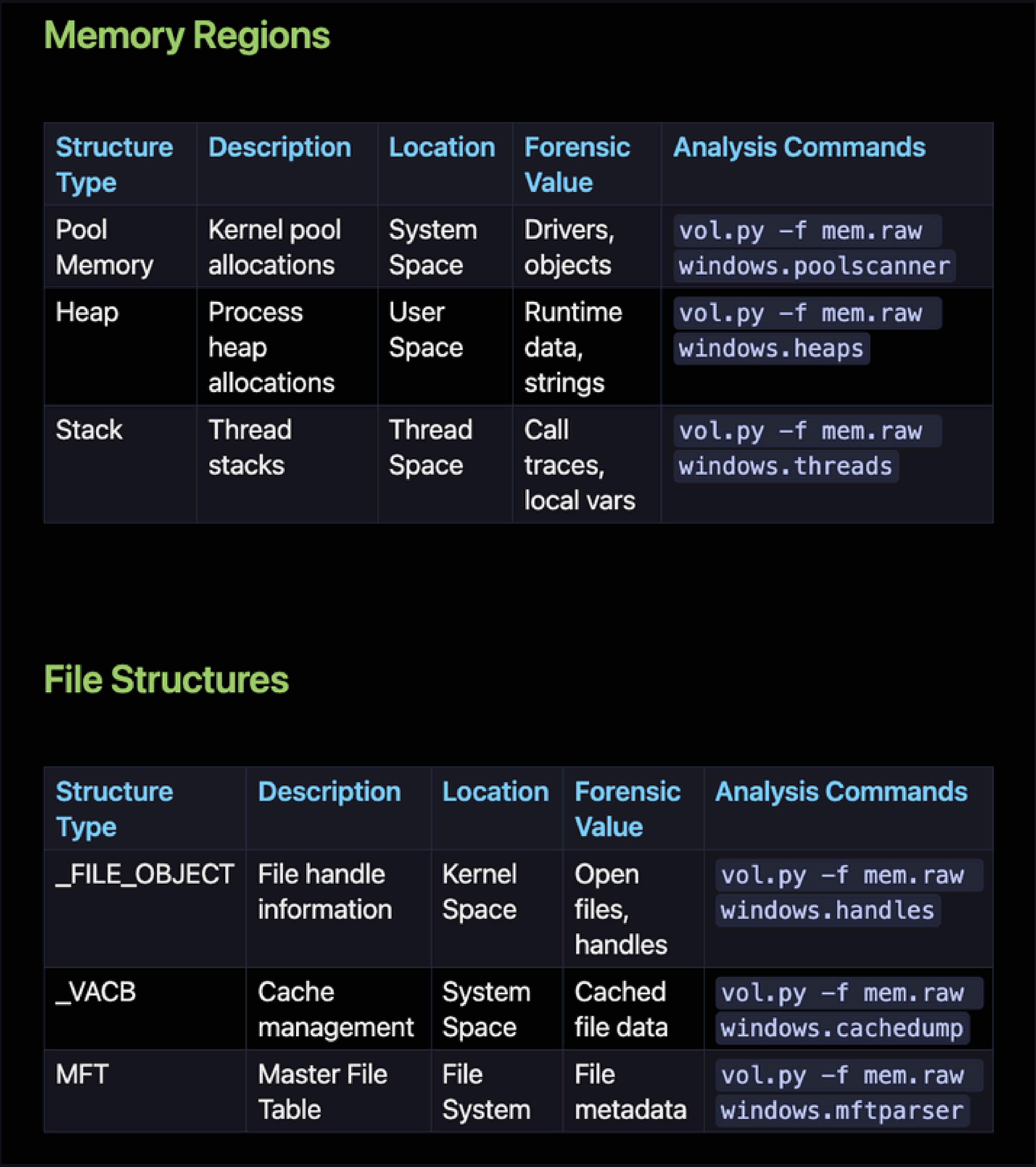
Memory Structure

Process Structures

Structure Type	Description	Location	Forensic Value	Analysis Commands
_EPROCESS	Process Environment Block	Kernel Space	Process details, threads, handles	`vol.py -f mem.raw windows.pslist`
_PEB	Process Environment Block	User Space	DLLs, env variables, cmdline	`vol.py -f mem.raw windows.dlllist`
VAD	Virtual Address Descriptor	Process Space	Memory mappings, injected code	`vol.py -f mem.raw windows.vadinfo`

Kernel Structures

Structure Type	Description	Location	Forensic Value	Analysis Commands
SSDT	System Service Descriptor Table	Kernel Space	Hooks, rootkit detection	`vol.py -f mem.raw windows.ssdt`
IDT	Interrupt Descriptor Table	Kernel Space	Interrupt handlers, hooks	`vol.py -f mem.raw windows.idt`
KPCR	Processor Control Region	Per CPU	CPU state, thread info	`vol.py -f mem.raw windows.kpcr`

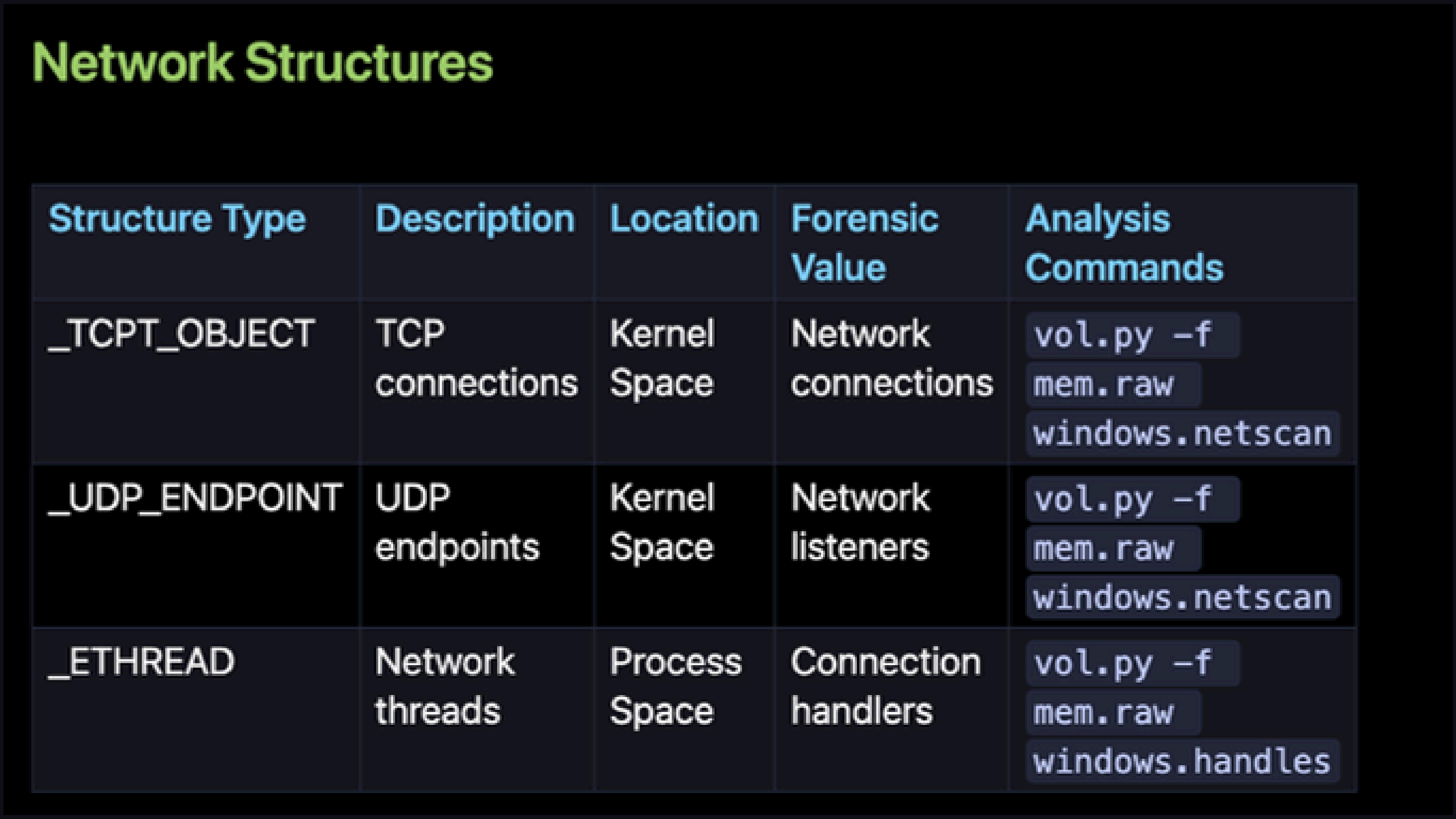


Memory Regions

Structure Type	Description	Location	Forensic Value	Analysis Commands
Pool Memory	Kernel pool allocations	System Space	Drivers, objects	`vol.py -f mem.raw windows.poolscanner`
Heap	Process heap allocations	User Space	Runtime data, strings	`vol.py -f mem.raw windows.heaps`
Stack	Thread stacks	Thread Space	Call traces, local vars	`vol.py -f mem.raw windows.threads`

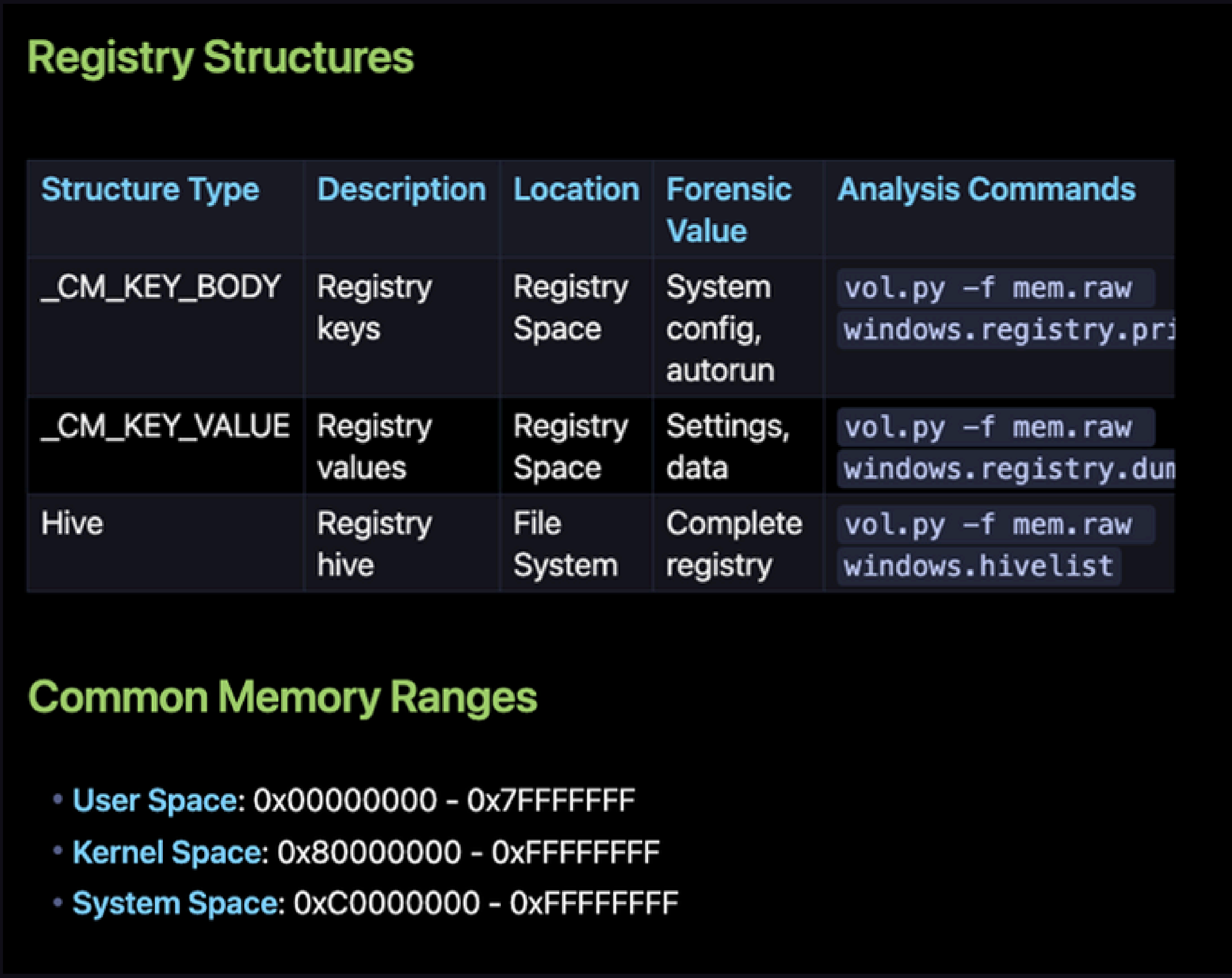
File Structures

Structure Type	Description	Location	Forensic Value	Analysis Commands
_FILE_OBJECT	File handle information	Kernel Space	Open files, handles	`vol.py -f mem.raw windows.handles`
_VACB	Cache management	System Space	Cached file data	`vol.py -f mem.raw windows.cachedump`
MFT	Master File Table	File System	File metadata	`vol.py -f mem.raw windows.mftparser`



Network Structures

Structure Type	Description	Location	Forensic Value	Analysis Commands
_TCPT_OBJECT	TCP connections	Kernel Space	Network connections	`vol.py -f mem.raw windows.netscan`
_UDP_ENDPOINT	UDP endpoints	Kernel Space	Network listeners	`vol.py -f mem.raw windows.netscan`
_ETHREAD	Network threads	Process Space	Connection handlers	`vol.py -f mem.raw windows.handles`

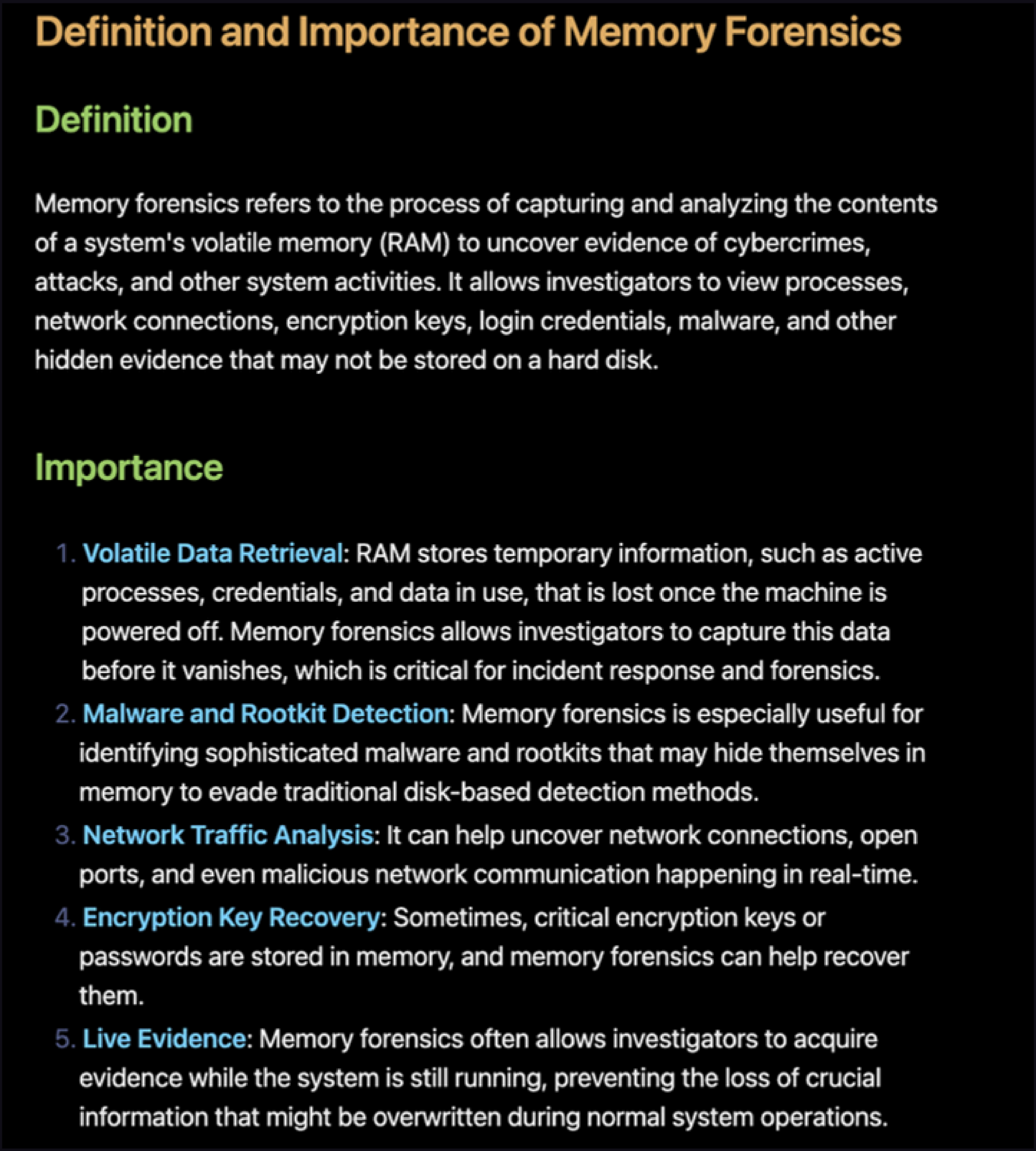


Registry Structures

Structure Type	Description	Location	Forensic Value	Analysis Commands
_CM_KEY_BODY	Registry keys	Registry Space	System config, autorun	`vol.py -f mem.raw windows.registry.pri`
_CM_KEY_VALUE	Registry values	Registry Space	Settings, data	`vol.py -f mem.raw windows.registry.dum`
Hive	Registry hive	File System	Complete registry	`vol.py -f mem.raw windows.hivelist`

Common Memory Ranges

- **User Space**: 0x00000000 - 0x7FFFFFFF
- **Kernel Space**: 0x80000000 - 0xFFFFFFFF
- **System Space**: 0xC0000000 - 0xFFFFFFFF



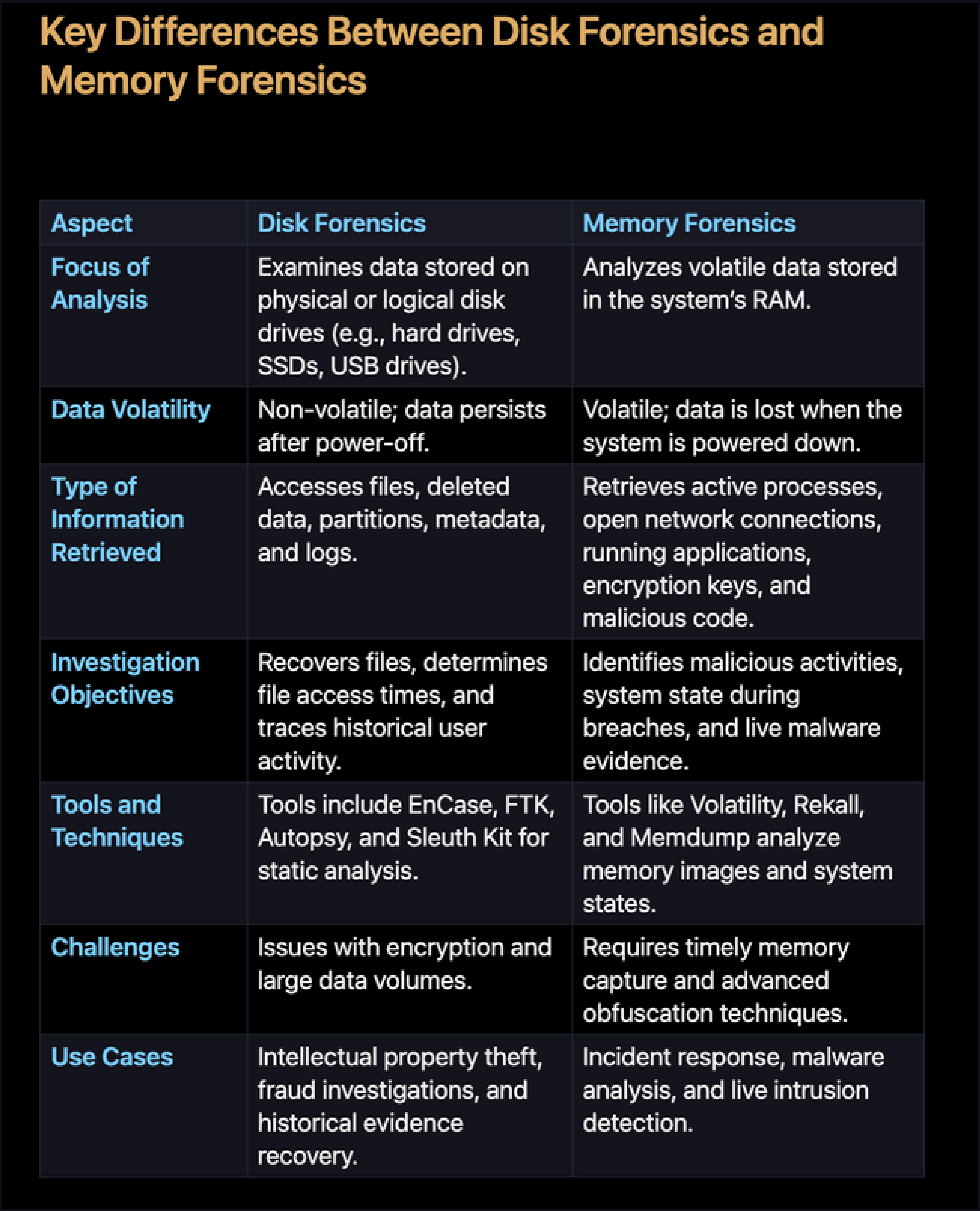
Definition and Importance of Memory Forensics

Definition

Memory forensics refers to the process of capturing and analyzing the contents of a system's volatile memory (RAM) to uncover evidence of cybercrimes, attacks, and other system activities. It allows investigators to view processes, network connections, encryption keys, login credentials, malware, and other hidden evidence that may not be stored on a hard disk.

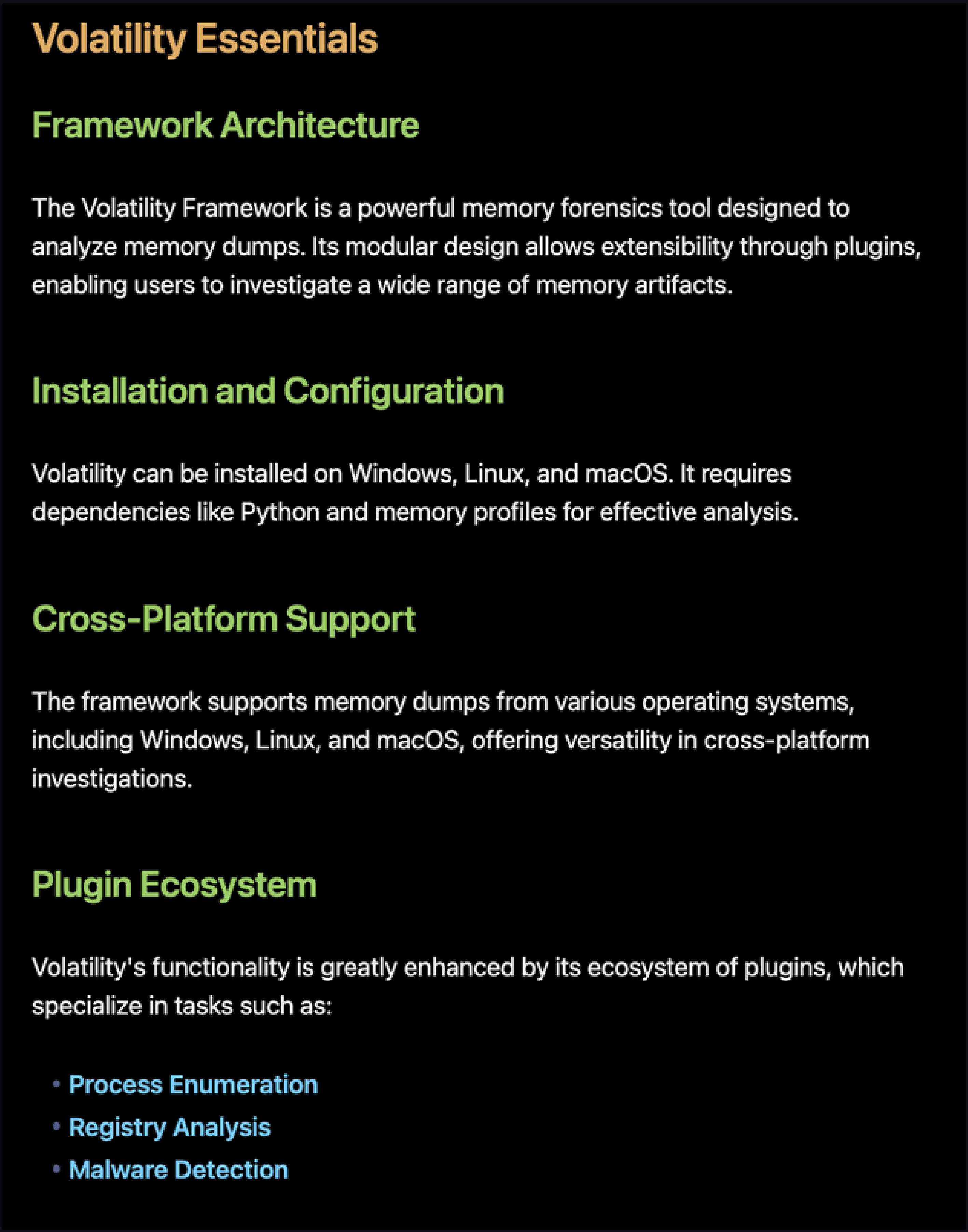
Importance

1. **Volatile Data Retrieval**: RAM stores temporary information, such as active processes, credentials, and data in use, that is lost once the machine is powered off. Memory forensics allows investigators to capture this data before it vanishes, which is critical for incident response and forensics.
2. **Malware and Rootkit Detection**: Memory forensics is especially useful for identifying sophisticated malware and rootkits that may hide themselves in memory to evade traditional disk-based detection methods.
3. **Network Traffic Analysis**: It can help uncover network connections, open ports, and even malicious network communication happening in real-time.
4. **Encryption Key Recovery**: Sometimes, critical encryption keys or passwords are stored in memory, and memory forensics can help recover them.
5. **Live Evidence**: Memory forensics often allows investigators to acquire evidence while the system is still running, preventing the loss of crucial information that might be overwritten during normal system operations.



Key Differences Between Disk Forensics and Memory Forensics

Aspect	Disk Forensics	Memory Forensics
Focus of Analysis	Examines data stored on physical or logical disk drives (e.g., hard drives, SSDs, USB drives).	Analyzes volatile data stored in the system's RAM.
Data Volatility	Non-volatile; data persists after power-off.	Volatile; data is lost when the system is powered down.
Type of Information Retrieved	Accesses files, deleted data, partitions, metadata, and logs.	Retrieves active processes, open network connections, running applications, encryption keys, and malicious code.
Investigation Objectives	Recovers files, determines file access times, and traces historical user activity.	Identifies malicious activities, system state during breaches, and live malware evidence.
Tools and Techniques	Tools include EnCase, FTK, Autopsy, and Sleuth Kit for static analysis.	Tools like Volatility, Rekall, and Memdump analyze memory images and system states.
Challenges	Issues with encryption and large data volumes.	Requires timely memory capture and advanced obfuscation techniques.
Use Cases	Intellectual property theft, fraud investigations, and historical evidence recovery.	Incident response, malware analysis, and live intrusion detection.



Volatility Essentials

Framework Architecture

The Volatility Framework is a powerful memory forensics tool designed to analyze memory dumps. Its modular design allows extensibility through plugins, enabling users to investigate a wide range of memory artifacts.

Installation and Configuration

Volatility can be installed on Windows, Linux, and macOS. It requires dependencies like Python and memory profiles for effective analysis.

Cross-Platform Support

The framework supports memory dumps from various operating systems, including Windows, Linux, and macOS, offering versatility in cross-platform investigations.

Plugin Ecosystem

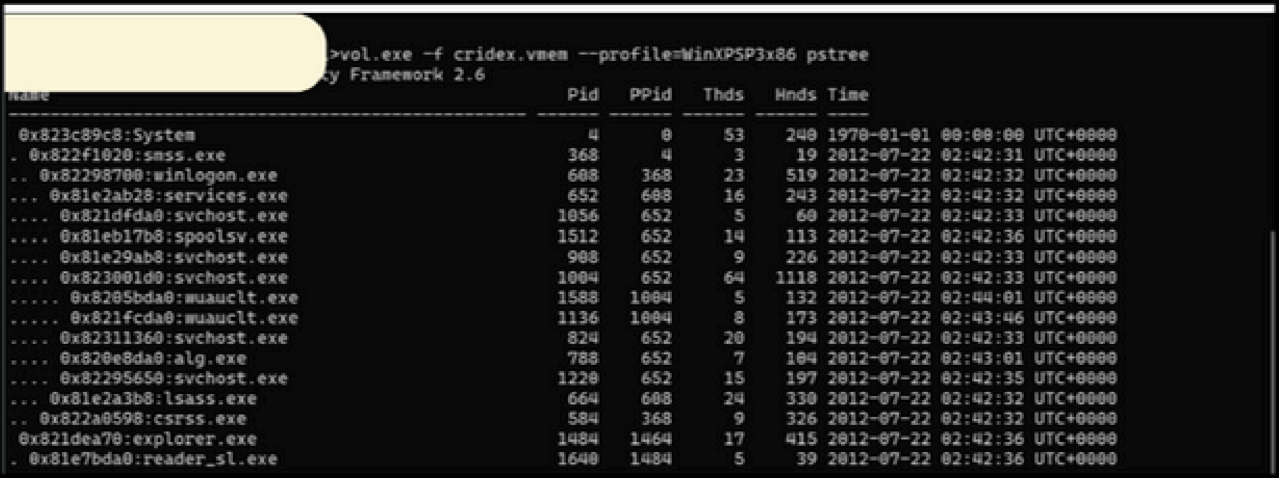
Volatility's functionality is greatly enhanced by its ecosystem of plugins, which specialize in tasks such as:

- Process Enumeration
- Registry Analysis
- Malware Detection

Memory Profile Selection

Accurate memory profile selection ensures the framework can correctly interpret the memory dump, matching it to the target system's kernel and configurations.

```
vol.exe -f cridex.vmem imageinfo
```



```
[redacted]>vol.exe -f cridex.vmem --profile=WinXPSP3x86 pstree
y Framework 2.6
Name                                                  Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----
 0x823c89c8:System                                      4      0     53    240 1970-01-01 00:00:00 UTC+0000
. 0x822f1020:smss.exe                                 368      4      3     19 2012-07-22 02:42:31 UTC+0000
.. 0x82298700:winlogon.exe                            608    368     23    519 2012-07-22 02:42:32 UTC+0000
... 0x81e2ab28:services.exe                           652    608     16    243 2012-07-22 02:42:32 UTC+0000
.... 0x821dfda0:svchost.exe                          1056    652      5     60 2012-07-22 02:42:33 UTC+0000
.... 0x81eb17b8:spoolsv.exe                          1512    652     14    113 2012-07-22 02:42:36 UTC+0000
.... 0x81e29ab8:svchost.exe                           908    652      9    226 2012-07-22 02:42:33 UTC+0000
.... 0x823001d0:svchost.exe                          1004    652     64   1118 2012-07-22 02:42:33 UTC+0000
..... 0x8205bda0:wuauclt.exe                         1588   1004      5    132 2012-07-22 02:44:01 UTC+0000
..... 0x821fcda0:wuauclt.exe                         1136   1004      8    173 2012-07-22 02:43:46 UTC+0000
.... 0x82311360:svchost.exe                           824    652     20    194 2012-07-22 02:42:33 UTC+0000
.... 0x820e8da0:alg.exe                               788    652      7    104 2012-07-22 02:43:01 UTC+0000
.... 0x82295650:svchost.exe                          1220    652     15    197 2012-07-22 02:42:35 UTC+0000
... 0x81e2a3b8:lsass.exe                              664    608     24    330 2012-07-22 02:42:32 UTC+0000
.. 0x822a0598:csrss.exe                               584    368      9    326 2012-07-22 02:42:32 UTC+0000
 0x821dea70:explorer.exe                             1484   1464     17    415 2012-07-22 02:42:36 UTC+0000
. 0x81e7bda0:reader_sl.exe                           1640   1484      5     39 2012-07-22 02:42:36 UTC+0000
```

Core Investigation Plugins

Plugins such as `pslist`, `psscan`, and `dlllist` form the backbone of forensic investigations. These tools provide detailed insights into:

- Active and hidden processes.
- Loaded libraries.
- Critical system components.

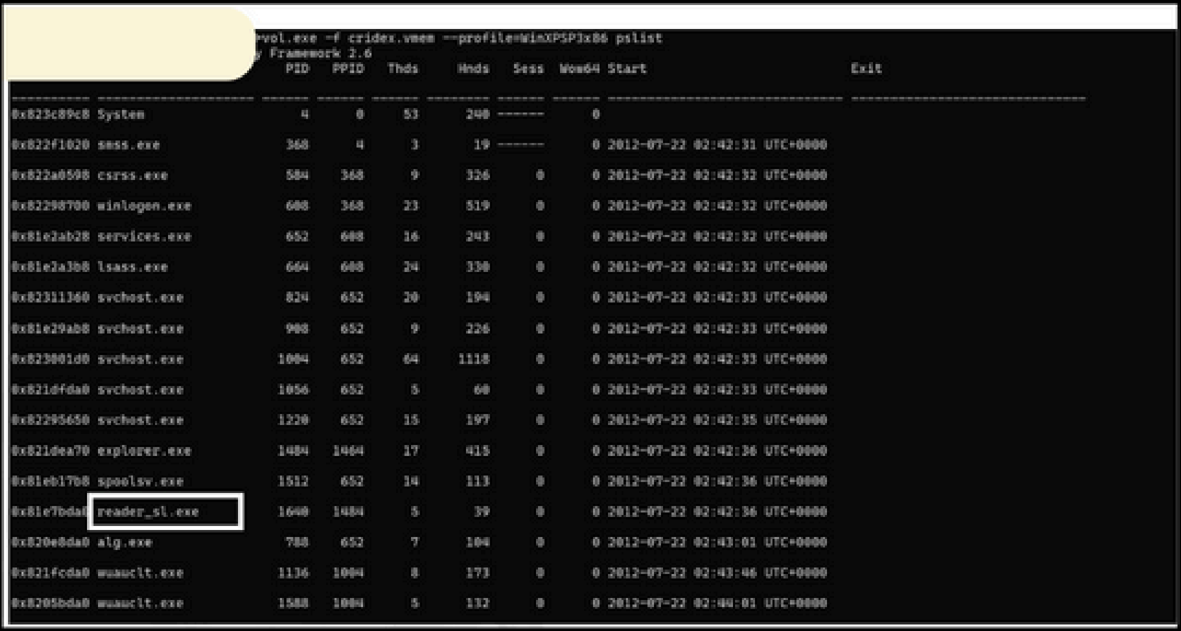
Key Benefits:

1. **Tracing Program Execution**: Enables analysts to map the lifecycle of processes and identify anomalies.
2. **Uncovering Suspicious Activity**: Helps detect hidden or malicious processes.
3. **Operational State Mapping**: Provides a snapshot of the system's active state for a comprehensive forensic analysis.

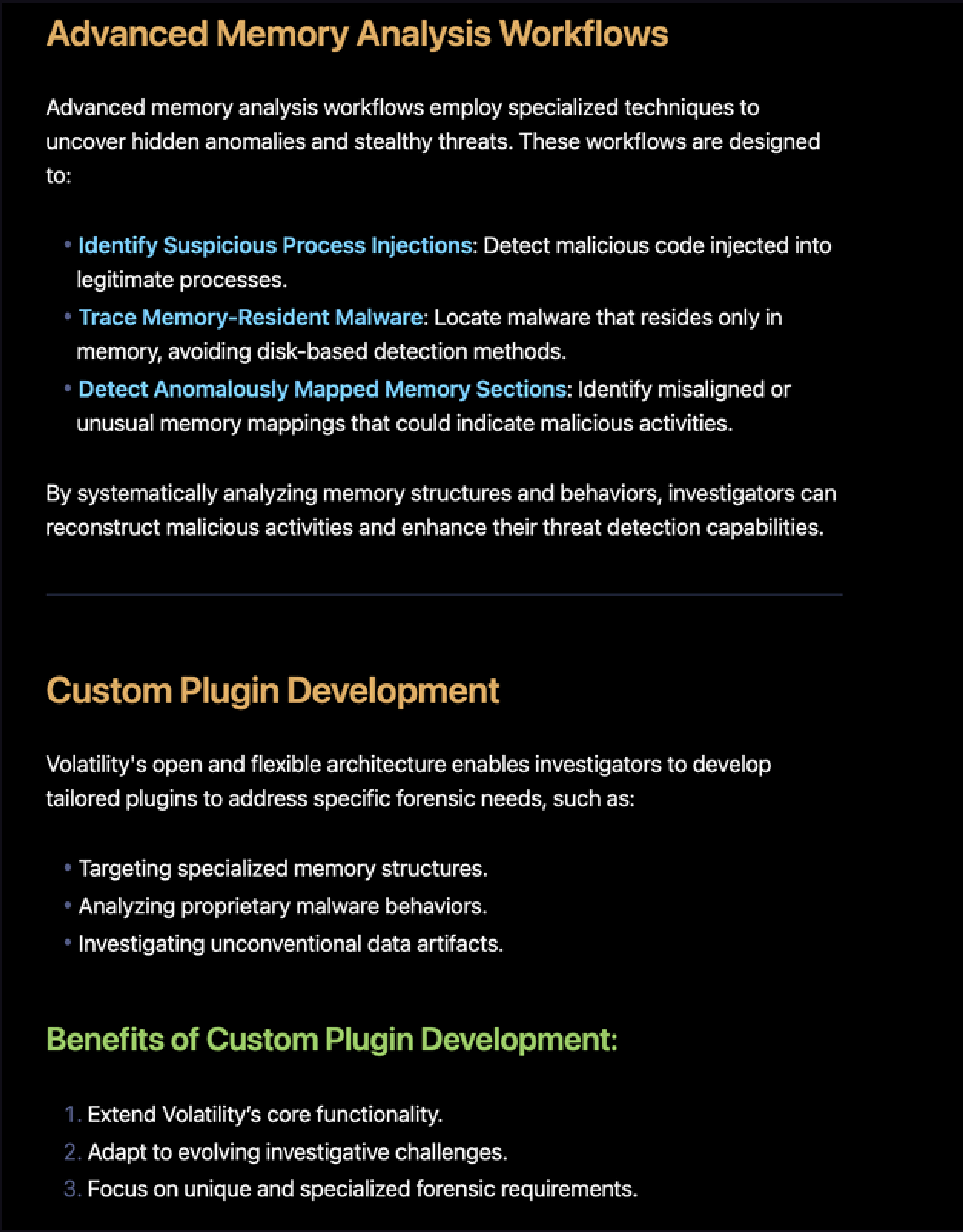
Example Command:

```
vol.exe -f cridex.vmem --profile=WinXPSP3x86 pslist
```

This command uses Volatility to analyze the memory dump (`cridex.vmem`) for a system with the specified profile (`WinXPSP3x86`) and lists active processes using the `pslist` plugin.



```
[redacted]>vol.exe -f cridex.vmem --profile=WinXPSP3x86 pslist
          y Framework 2.6
                                PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
---------- -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ------------------------------
0x823c89c8 System                    4      0     53      240 ------      0
0x822f1020 smss.exe                368      4      3       19 ------      0 2012-07-22 02:42:31 UTC+0000
0x822a0598 csrss.exe               584    368      9      326      0      0 2012-07-22 02:42:32 UTC+0000
0x82298700 winlogon.exe            608    368     23      519      0      0 2012-07-22 02:42:32 UTC+0000
0x81e2ab28 services.exe            652    608     16      243      0      0 2012-07-22 02:42:32 UTC+0000
0x81e2a3b8 lsass.exe               664    608     24      330      0      0 2012-07-22 02:42:32 UTC+0000
0x82311360 svchost.exe             824    652     20      194      0      0 2012-07-22 02:42:33 UTC+0000
0x81e29ab8 svchost.exe             908    652      9      226      0      0 2012-07-22 02:42:33 UTC+0000
0x823001d0 svchost.exe            1004    652     64     1118      0      0 2012-07-22 02:42:33 UTC+0000
0x821dfda0 svchost.exe            1056    652      5       60      0      0 2012-07-22 02:42:33 UTC+0000
0x82295650 svchost.exe            1220    652     15      197      0      0 2012-07-22 02:42:35 UTC+0000
0x821dea70 explorer.exe           1484   1464     17      415      0      0 2012-07-22 02:42:36 UTC+0000
0x81eb17b8 spoolsv.exe            1512    652     14      113      0      0 2012-07-22 02:42:36 UTC+0000
0x81e7bda0 reader_sl.exe          1640   1484      5       39      0      0 2012-07-22 02:42:36 UTC+0000
0x820e8da0 alg.exe                 788    652      7      104      0      0 2012-07-22 02:43:01 UTC+0000
0x821fcda0 wuauclt.exe            1136   1004      8      173      0      0 2012-07-22 02:43:46 UTC+0000
0x8205bda0 wuauclt.exe            1588   1004      5      132      0      0 2012-07-22 02:44:01 UTC+0000
```



Advanced Memory Analysis Workflows

Advanced memory analysis workflows employ specialized techniques to uncover hidden anomalies and stealthy threats. These workflows are designed to:

- **Identify Suspicious Process Injections**: Detect malicious code injected into legitimate processes.
- **Trace Memory-Resident Malware**: Locate malware that resides only in memory, avoiding disk-based detection methods.
- **Detect Anomalously Mapped Memory Sections**: Identify misaligned or unusual memory mappings that could indicate malicious activities.

By systematically analyzing memory structures and behaviors, investigators can reconstruct malicious activities and enhance their threat detection capabilities.

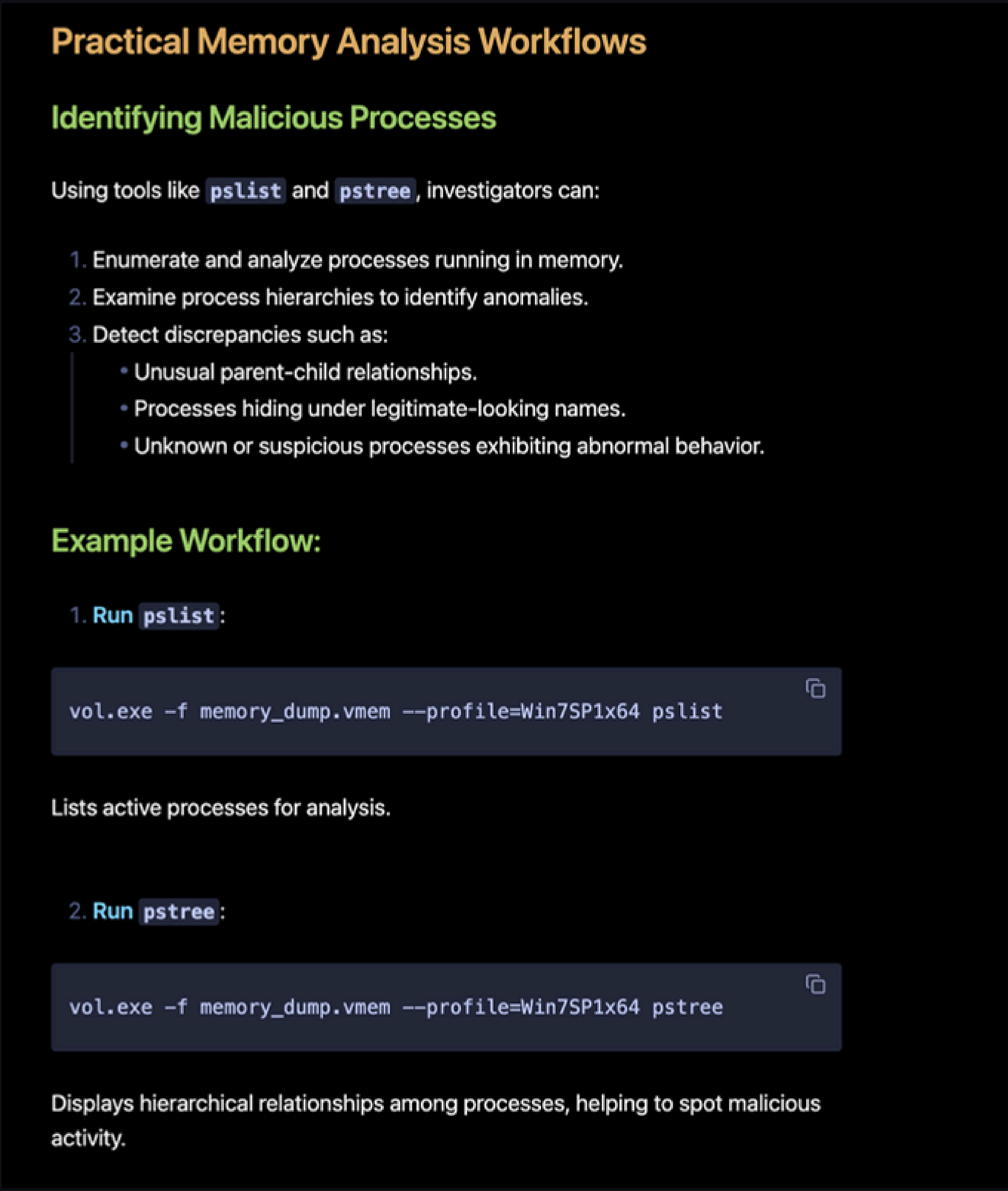
Custom Plugin Development

Volatility's open and flexible architecture enables investigators to develop tailored plugins to address specific forensic needs, such as:

- Targeting specialized memory structures.
- Analyzing proprietary malware behaviors.
- Investigating unconventional data artifacts.

Benefits of Custom Plugin Development:

1. Extend Volatility's core functionality.
2. Adapt to evolving investigative challenges.
3. Focus on unique and specialized forensic requirements.



Practical Memory Analysis Workflows

Identifying Malicious Processes

Using tools like `pslist` and `pstree`, investigators can:

1. Enumerate and analyze processes running in memory.
2. Examine process hierarchies to identify anomalies.
3. Detect discrepancies such as:
   - Unusual parent-child relationships.
   - Processes hiding under legitimate-looking names.
   - Unknown or suspicious processes exhibiting abnormal behavior.

Example Workflow:

1. Run `pslist`:

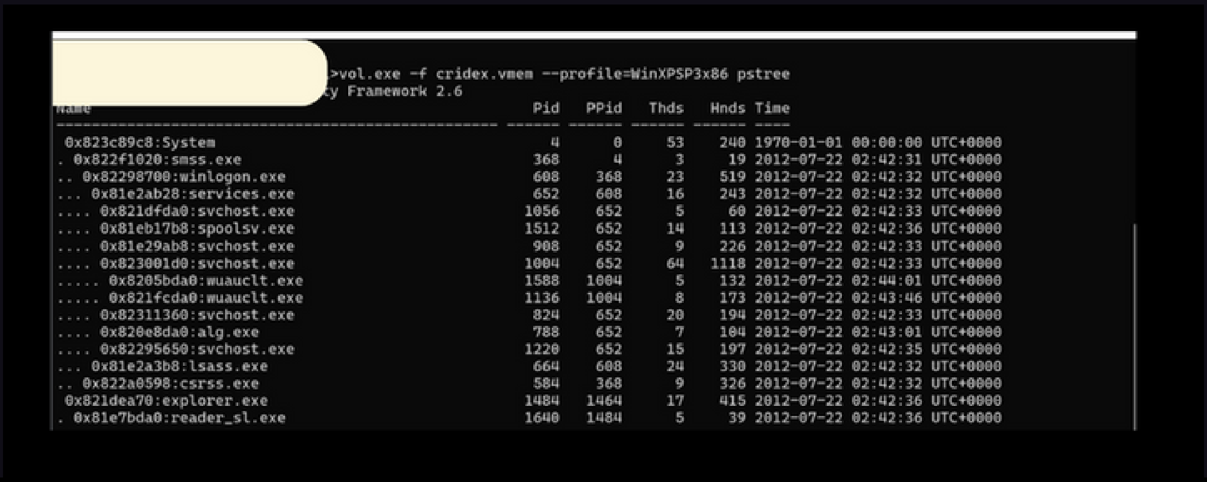
```
vol.exe -f memory_dump.vmem --profile=Win7SP1x64 pslist
```

Lists active processes for analysis.

2. Run `pstree`:

```
vol.exe -f memory_dump.vmem --profile=Win7SP1x64 pstree
```

Displays hierarchical relationships among processes, helping to spot malicious activity.



```
[redacted]>vol.exe -f cridex.vmem --profile=WinXPSP3x86 pstree
          ty Framework 2.6
Name                                                  Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----
 0x823c89c8:System                                      4      0     53    240 1970-01-01 00:00:00 UTC+0000
. 0x822f1020:smss.exe                                 368      4      3     19 2012-07-22 02:42:31 UTC+0000
.. 0x82298700:winlogon.exe                            608    368     23    519 2012-07-22 02:42:32 UTC+0000
... 0x81e2ab28:services.exe                           652    608     16    243 2012-07-22 02:42:32 UTC+0000
.... 0x821dfda0:svchost.exe                          1056    652      5     60 2012-07-22 02:42:33 UTC+0000
.... 0x81eb17b8:spoolsv.exe                          1512    652     14    113 2012-07-22 02:42:36 UTC+0000
.... 0x81e29ab8:svchost.exe                           908    652      9    226 2012-07-22 02:42:33 UTC+0000
.... 0x823001d0:svchost.exe                          1004    652     64   1118 2012-07-22 02:42:33 UTC+0000
..... 0x8205bda0:wuauclt.exe                         1588   1004      5    132 2012-07-22 02:44:01 UTC+0000
..... 0x821fcda0:wuauclt.exe                         1136   1004      8    173 2012-07-22 02:43:46 UTC+0000
.... 0x82311360:svchost.exe                           824    652     20    194 2012-07-22 02:42:33 UTC+0000
.... 0x820e8da0:alg.exe                               788    652      7    104 2012-07-22 02:43:01 UTC+0000
.... 0x82295650:svchost.exe                          1220    652     15    197 2012-07-22 02:42:35 UTC+0000
... 0x81e2a3b8:lsass.exe                              664    608     24    330 2012-07-22 02:42:32 UTC+0000
.. 0x822a0598:csrss.exe                               584    368      9    326 2012-07-22 02:42:32 UTC+0000
 0x821dea70:explorer.exe                             1484   1464     17    415 2012-07-22 02:42:36 UTC+0000
. 0x81e7bda0:reader_sl.exe                           1640   1484      5     39 2012-07-22 02:42:36 UTC+0000
```

Investigation of Running Processes

Investigating running processes is a crucial step in memory forensics, particularly when analyzing for potential malware, such as Cridex. This section outlines the approach to identifying suspicious processes using Volatility.

Steps for Investigating Running Processes

1. Checking for Suspicious Process Names

Malware often disguises itself under legitimate-sounding process names. A detailed inspection of the process list can help uncover anomalies.

2. Checking for Processes with Different Parent Process IDs (PPID)

Processes with unexpected or unusual parent process IDs can indicate tampering or injection by malicious actors. Analyzing the PPID relationships provides critical clues.

Case Example: Identifying `reader_sl.exe`

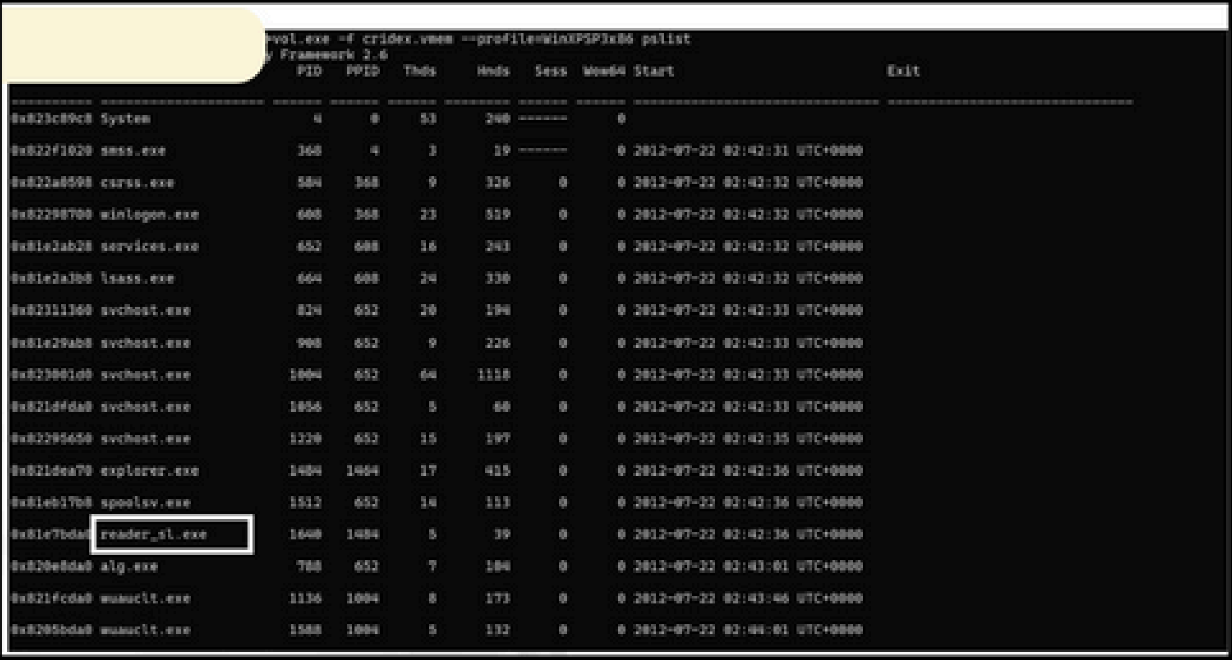
In this investigation, the malware is disguised under the process name `reader_sl.exe`.

Command for Analysis

To inspect the running processes and identify suspicious entries like `reader_sl.exe`, use the following Volatility command:

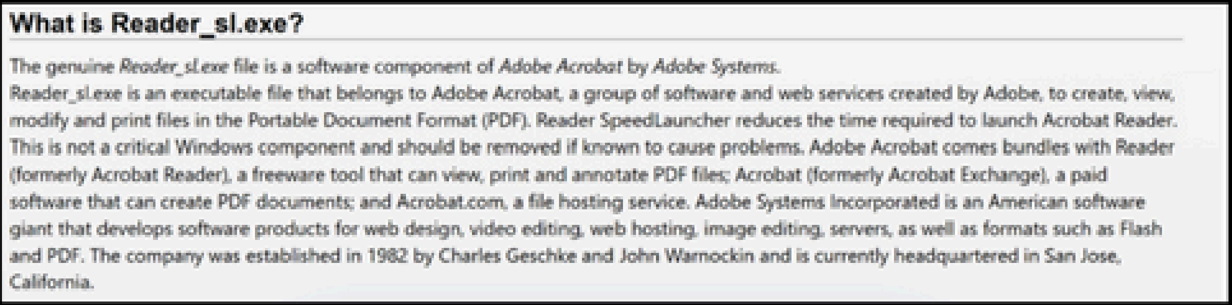
```
vol.exe -f cridex.vmem --profile=WinXPSP3x86 pslist
```

- `-f cridex.vmem`: Specifies the memory dump file to analyze.
- `--profile=WinXPSP3x86`: Defines the memory profile matching the target system (Windows XP SP3 x86).
- `pslist`: Lists all running processes and their parent-child relationships.



```
                                 >vol.exe -f cridex.vmem --profile=WinXPSP3x86 pslist
                                 y Framework 2.6
                                   PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
---------- -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ------------------------------
0x823c89c8 System                    4      0     53      240 ------      0
0x822f1020 smss.exe                368      4      3       19 ------      0 2012-07-22 02:42:31 UTC+0000
0x822a0598 csrss.exe               584    368      9      326      0      0 2012-07-22 02:42:32 UTC+0000
0x82298700 winlogon.exe            608    368     23      519      0      0 2012-07-22 02:42:32 UTC+0000
0x81e2ab28 services.exe            652    608     16      243      0      0 2012-07-22 02:42:32 UTC+0000
0x81e2a3b8 lsass.exe               664    608     24      330      0      0 2012-07-22 02:42:32 UTC+0000
0x82311360 svchost.exe             824    652     20      194      0      0 2012-07-22 02:42:33 UTC+0000
0x81e29ab8 svchost.exe             908    652      9      226      0      0 2012-07-22 02:42:33 UTC+0000
0x823001d0 svchost.exe            1004    652     64     1118      0      0 2012-07-22 02:42:33 UTC+0000
0x821dfda0 svchost.exe            1056    652      5       60      0      0 2012-07-22 02:42:33 UTC+0000
0x82295650 svchost.exe            1220    652     15      197      0      0 2012-07-22 02:42:35 UTC+0000
0x821dea70 explorer.exe           1484   1464     17      415      0      0 2012-07-22 02:42:36 UTC+0000
0x81eb17b8 spoolsv.exe            1512    652     14      113      0      0 2012-07-22 02:42:36 UTC+0000
0x81e7bda0 reader_sl.exe          1640   1484      5       39      0      0 2012-07-22 02:42:36 UTC+0000
0x820e8da0 alg.exe                 788    652      7      104      0      0 2012-07-22 02:43:01 UTC+0000
0x821fcda0 wuauclt.exe            1136   1004      8      173      0      0 2012-07-22 02:43:46 UTC+0000
0x8205bda0 wuauclt.exe            1588   1004      5      132      0      0 2012-07-22 02:44:01 UTC+0000
```

While selecting the suspicious process name we will have to know what is the process functionality in addition what is the purpose of this suspicious process .



What is Reader_sl.exe?

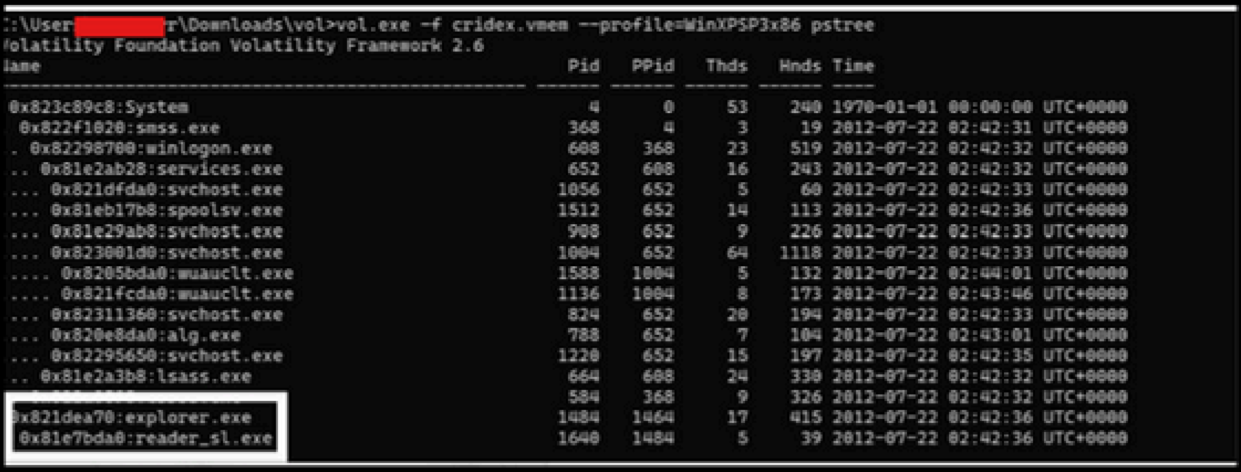
The genuine *Reader_sl.exe* file is a software component of *Adobe Acrobat* by *Adobe Systems*.
Reader_sl.exe is an executable file that belongs to Adobe Acrobat, a group of software and web services created by Adobe, to create, view, modify and print files in the Portable Document Format (PDF). Reader SpeedLauncher reduces the time required to launch Acrobat Reader. This is not a critical Windows component and should be removed if known to cause problems. Adobe Acrobat comes bundles with Reader (formerly Acrobat Reader), a freeware tool that can view, print and annotate PDF files; Acrobat (formerly Acrobat Exchange), a paid software that can create PDF documents; and Acrobat.com, a file hosting service. Adobe Systems Incorporated is an American software giant that develops software products for web design, video editing, web hosting, image editing, servers, as well as formats such as Flash and PDF. The company was established in 1982 by Charles Geschke and John Warnockin and is currently headquartered in San Jose, California.

Based on the search, it is suspected that the infected host machine may have been compromised by malicious documents, such as `.pdf` or `.docx`.

Tracing the Creator of `reader_sl.exe`

Another plugin from Volatility, `pstree`, can be used to identify which process or program created `reader_sl.exe`. This analysis can provide additional indicators for investigation.

```
vol.exe -f cridex.vmem --profile=WinXPSP3x86 pstree
```



```
:\User[redacted]r\Downloads\vol>vol.exe -f cridex.vmem --profile=WinXPSP3x86 pstree
olatility Foundation Volatility Framework 2.6
ame                                                  Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----
0x823c89c8:System                                      4      0     53    240 1970-01-01 00:00:00 UTC+0000
 0x822f1020:smss.exe                                 368      4      3     19 2012-07-22 02:42:31 UTC+0000
. 0x82298700:winlogon.exe                            608    368     23    519 2012-07-22 02:42:32 UTC+0000
.. 0x81e2ab28:services.exe                           652    608     16    243 2012-07-22 02:42:32 UTC+0000
... 0x821dfda0:svchost.exe                          1056    652      5     60 2012-07-22 02:42:33 UTC+0000
... 0x81eb17b8:spoolsv.exe                          1512    652     14    113 2012-07-22 02:42:36 UTC+0000
... 0x81e29ab8:svchost.exe                           908    652      9    226 2012-07-22 02:42:33 UTC+0000
... 0x823001d0:svchost.exe                          1004    652     64   1118 2012-07-22 02:42:33 UTC+0000
.... 0x8205bda0:wuauclt.exe                         1588   1004      5    132 2012-07-22 02:44:01 UTC+0000
.... 0x821fcda0:wuauclt.exe                         1136   1004      8    173 2012-07-22 02:43:46 UTC+0000
... 0x82311360:svchost.exe                           824    652     20    194 2012-07-22 02:42:33 UTC+0000
... 0x820e8da0:alg.exe                               788    652      7    104 2012-07-22 02:43:01 UTC+0000
... 0x82295650:svchost.exe                          1220    652     15    197 2012-07-22 02:42:35 UTC+0000
.. 0x81e2a3b8:lsass.exe                              664    608     24    330 2012-07-22 02:42:32 UTC+0000
                                                     584    368      9    326 2012-07-22 02:42:32 UTC+0000
x821dea70:explorer.exe                              1484   1464     17    415 2012-07-22 02:42:36 UTC+0000
 0x81e7bda0:reader_sl.exe                           1640   1484      5     39 2012-07-22 02:42:36 UTC+0000
```

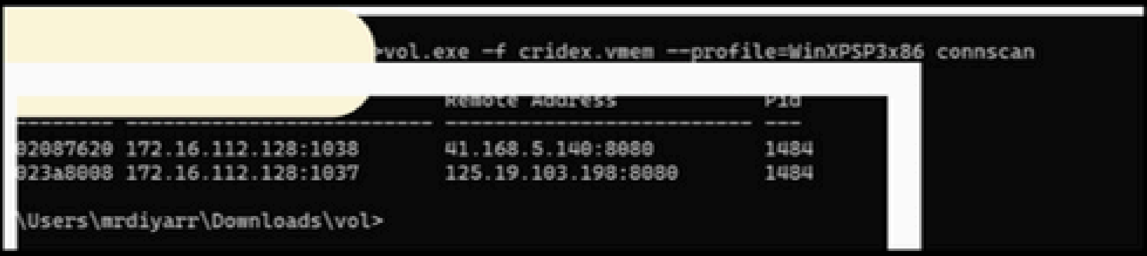
Analysis Using `pstree` Plugin

Based on the details provided by the `pstree` plugin, we have a clue that `explorer.exe` is creating `reader_sl.exe`. This potentially indicates that the infected host machine opened malicious documents, such as those received from an attacker.

Investigating a Process's Internet Connection

Investigating a process's internet connection in memory forensics is crucial for identifying potential malicious activities, such as communication with Command and Control (C&C) servers.

```
vol.exe -f cridex.vmem --profile=WinXPSP3x86 connscan
```



```
>vol.exe -f cridex.vmem --profile=WinXPSP3x86 connscan

                                   Remote Address            Pid
-------- ------------------------- ------------------------- ---
02087620 172.16.112.128:1038       41.168.5.140:8080         1484
023a8008 172.16.112.128:1037       125.19.103.198:8080       1484

\Users\mrdiyarr\Downloads\vol>
```

Processes with Remote Connections

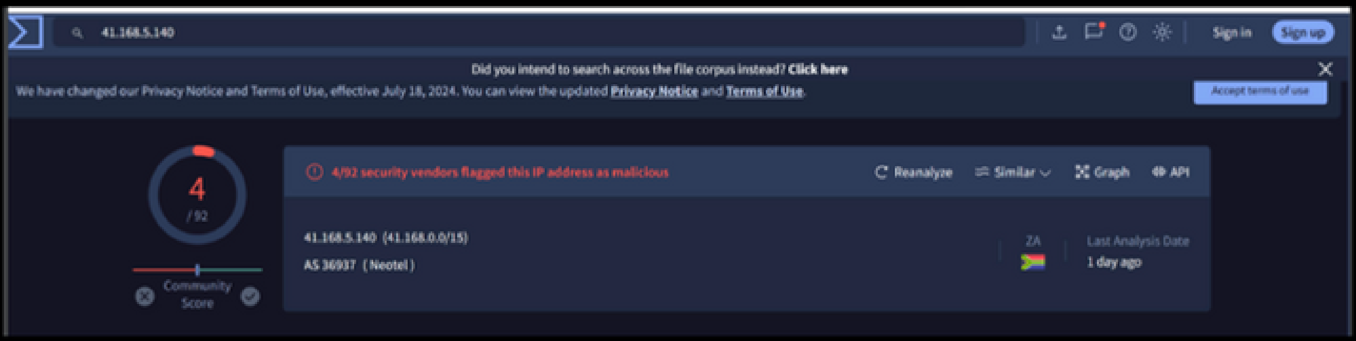
During the analysis, two processes were identified as having made a connection to remote addresses:

1. explorer.exe
2. Suspected process: reader_sl.exe

Logical Analysis

A logical question arises: **Why should `reader_sl.exe` establish an internet connection?**

To further investigate, it is essential to check the remote IP address associated with this connection on VirusTotal or similar threat intelligence platforms for potential malicious activity.



41.168.5.140

Did you intend to search across the file corpus instead? **Click here**

We have changed our Privacy Notice and Terms of Use, effective July 18, 2024. You can view the updated Privacy Notice and Terms of Use.

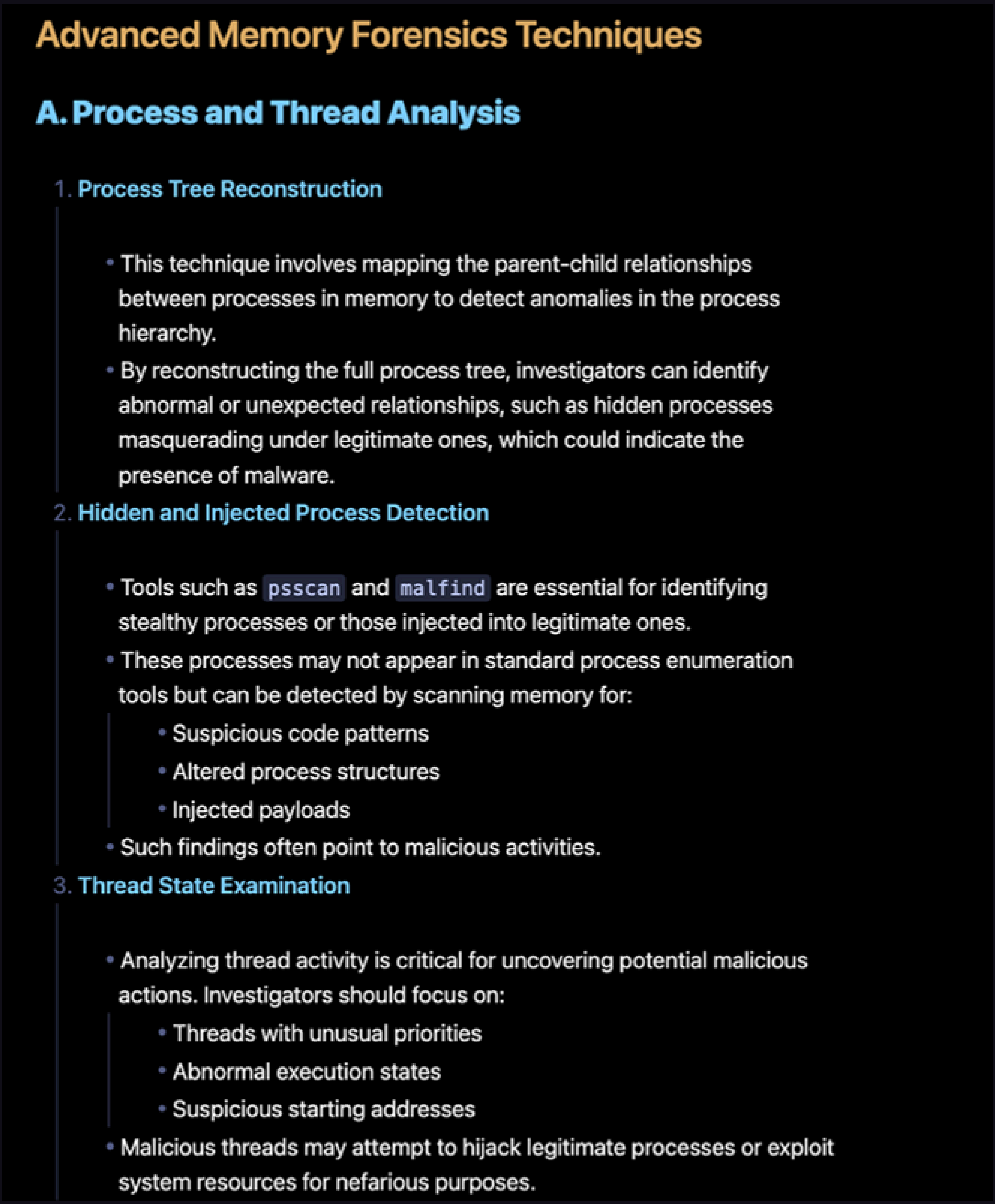
4/92 security vendors flagged this IP address as malicious

41.168.5.140 (41.168.0.0/15)
AS 36937 (Neotel)

ZA

Last Analysis Date
1 day ago

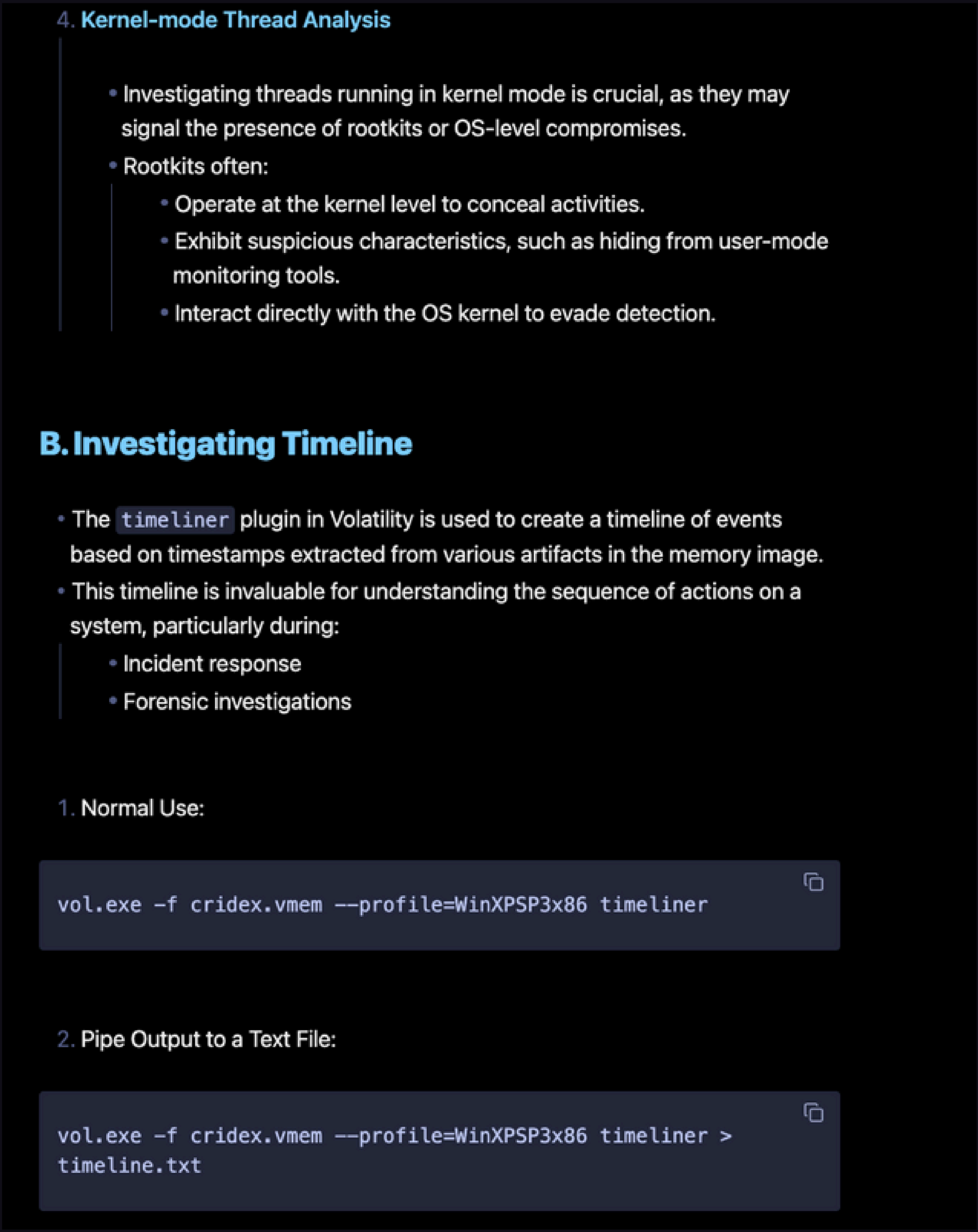
- **Parent Process ID (PPID):** `1484` (This is the PPID for `reader_sl.exe`.)



Advanced Memory Forensics Techniques

A. Process and Thread Analysis

1. Process Tree Reconstruction
   - This technique involves mapping the parent-child relationships between processes in memory to detect anomalies in the process hierarchy.
   - By reconstructing the full process tree, investigators can identify abnormal or unexpected relationships, such as hidden processes masquerading under legitimate ones, which could indicate the presence of malware.
2. Hidden and Injected Process Detection
   - Tools such as `psscan` and `malfind` are essential for identifying stealthy processes or those injected into legitimate ones.
   - These processes may not appear in standard process enumeration tools but can be detected by scanning memory for:
     - Suspicious code patterns
     - Altered process structures
     - Injected payloads
   - Such findings often point to malicious activities.
3. Thread State Examination
   - Analyzing thread activity is critical for uncovering potential malicious actions. Investigators should focus on:
     - Threads with unusual priorities
     - Abnormal execution states
     - Suspicious starting addresses
   - Malicious threads may attempt to hijack legitimate processes or exploit system resources for nefarious purposes.



4. **Kernel-mode Thread Analysis**

    - Investigating threads running in kernel mode is crucial, as they may signal the presence of rootkits or OS-level compromises.
    - Rootkits often:
        - Operate at the kernel level to conceal activities.
        - Exhibit suspicious characteristics, such as hiding from user-mode monitoring tools.
        - Interact directly with the OS kernel to evade detection.

B. Investigating Timeline

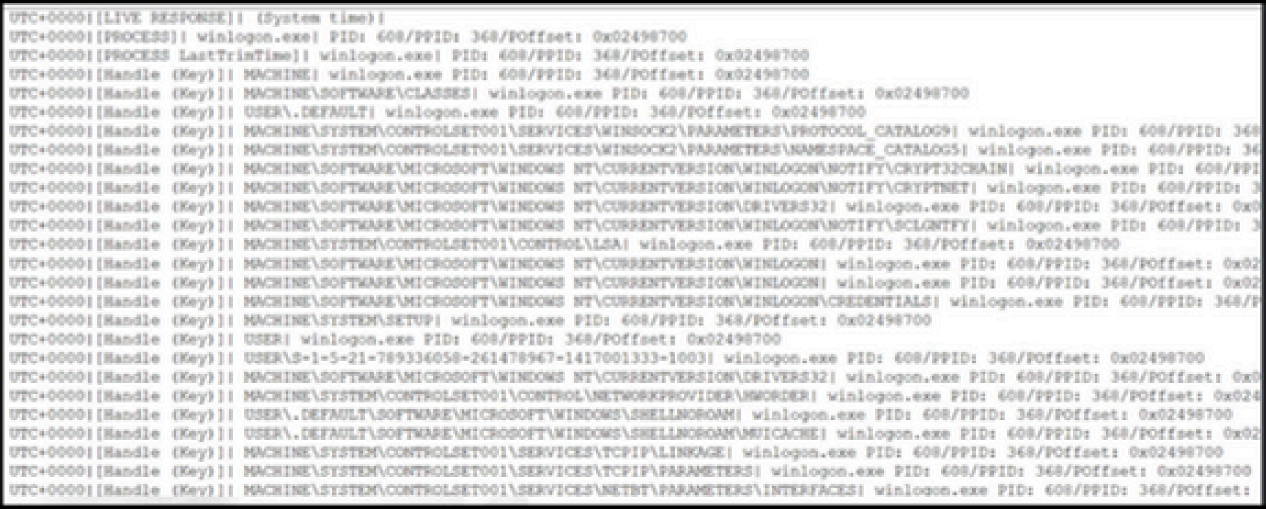
- The `timeliner` plugin in Volatility is used to create a timeline of events based on timestamps extracted from various artifacts in the memory image.
- This timeline is invaluable for understanding the sequence of actions on a system, particularly during:
    - Incident response
    - Forensic investigations

1. Normal Use:

```
vol.exe -f cridex.vmem --profile=WinXPSP3x86 timeliner
```

2. Pipe Output to a Text File:

```
vol.exe -f cridex.vmem --profile=WinXPSP3x86 timeliner >
timeline.txt
```



```
UTC+0000|[LIVE RESPONSE]|| (System time)|
UTC+0000|[PROCESS]|| winlogon.exe| PID: 608/PPID: 368/POffset: 0x02498700
UTC+0000|[PROCESS LastTrimTime]|| winlogon.exe| PID: 608/PPID: 368/POffset: 0x02498700
UTC+0000|[Handle (Key)]|| MACHINE| winlogon.exe PID: 608/PPID: 368/POffset: 0x02498700
UTC+0000|[Handle (Key)]|| MACHINE\SOFTWARE\CLASSES| winlogon.exe PID: 608/PPID: 368/POffset: 0x02498700
UTC+0000|[Handle (Key)]|| USER\.DEFAULT| winlogon.exe PID: 608/PPID: 368/POffset: 0x02498700
UTC+0000|[Handle (Key)]|| MACHINE\SYSTEM\CONTROLSET001\SERVICES\WINSOCK2\PARAMETERS\PROTOCOL_CATALOG9| winlogon.exe PID: 608/PPID: 368
UTC+0000|[Handle (Key)]|| MACHINE\SYSTEM\CONTROLSET001\SERVICES\WINSOCK2\PARAMETERS\NAMESPACE_CATALOG5| winlogon.exe PID: 608/PPID: 36
UTC+0000|[Handle (Key)]|| MACHINE\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\WINLOGON\NOTIFY\CRYPT32CHAIN| winlogon.exe PID: 608/PPI
UTC+0000|[Handle (Key)]|| MACHINE\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\WINLOGON\NOTIFY\CRYPTNET| winlogon.exe PID: 608/PPID: 3
UTC+0000|[Handle (Key)]|| MACHINE\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\DRIVERS32| winlogon.exe PID: 608/PPID: 368/POffset: 0x0
UTC+0000|[Handle (Key)]|| MACHINE\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\WINLOGON\NOTIFY\SCLGNTFY| winlogon.exe PID: 608/PPID: 3
UTC+0000|[Handle (Key)]|| MACHINE\SYSTEM\CONTROLSET001\CONTROL\LSA| winlogon.exe PID: 608/PPID: 368/POffset: 0x02498700
UTC+0000|[Handle (Key)]|| MACHINE\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\WINLOGON| winlogon.exe PID: 608/PPID: 368/POffset: 0x02
UTC+0000|[Handle (Key)]|| MACHINE\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\WINLOGON| winlogon.exe PID: 608/PPID: 368/POffset: 0x02
UTC+0000|[Handle (Key)]|| MACHINE\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\WINLOGON\CREDENTIALS| winlogon.exe PID: 608/PPID: 368/P
UTC+0000|[Handle (Key)]|| MACHINE\SYSTEM\SETUP| winlogon.exe PID: 608/PPID: 368/POffset: 0x02498700
UTC+0000|[Handle (Key)]|| USER| winlogon.exe PID: 608/PPID: 368/POffset: 0x02498700
UTC+0000|[Handle (Key)]|| USER\S-1-5-21-789336058-261478967-1417001333-1003| winlogon.exe PID: 608/PPID: 368/POffset: 0x02498700
UTC+0000|[Handle (Key)]|| MACHINE\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\DRIVERS32| winlogon.exe PID: 608/PPID: 368/POffset: 0x0
UTC+0000|[Handle (Key)]|| MACHINE\SYSTEM\CONTROLSET001\CONTROL\NETWORKPROVIDER\HWORDER| winlogon.exe PID: 608/PPID: 368/POffset: 0x024
UTC+0000|[Handle (Key)]|| USER\.DEFAULT\SOFTWARE\MICROSOFT\WINDOWS\SHELLNOROAM| winlogon.exe PID: 608/PPID: 368/POffset: 0x02498700
UTC+0000|[Handle (Key)]|| USER\.DEFAULT\SOFTWARE\MICROSOFT\WINDOWS\SHELLNOROAM\MUICACHE| winlogon.exe PID: 608/PPID: 368/POffset: 0x02
UTC+0000|[Handle (Key)]|| MACHINE\SYSTEM\CONTROLSET001\SERVICES\TCPIP\LINKAGE| winlogon.exe PID: 608/PPID: 368/POffset: 0x02498700
UTC+0000|[Handle (Key)]|| MACHINE\SYSTEM\CONTROLSET001\SERVICES\TCPIP\PARAMETERS| winlogon.exe PID: 608/PPID: 368/POffset: 0x02498700
UTC+0000|[Handle (Key)]|| MACHINE\SYSTEM\CONTROLSET001\SERVICES\NETBT\PARAMETERS\INTERFACES| winlogon.exe PID: 608/PPID: 368/POffset:
```

Investigation: Clipboard Hooking

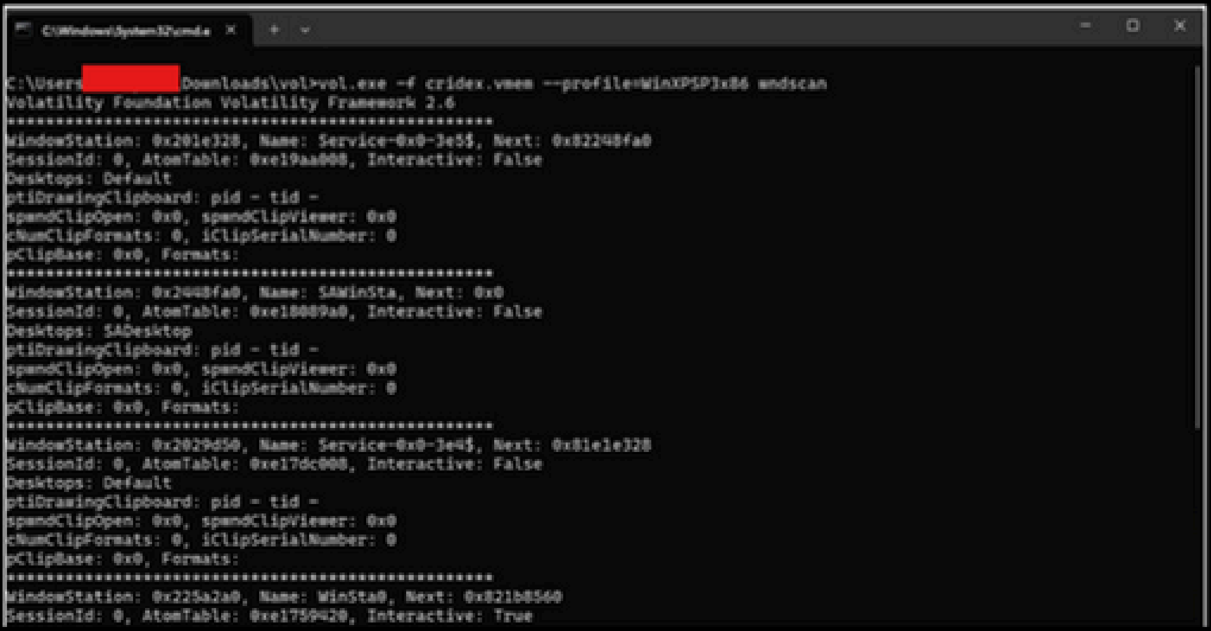
- The `wndscan` plugin in Volatility is utilized to scan for window objects in memory.
- This functionality is particularly useful for identifying both visible and hidden windows created by processes, which may include:
  - Malware-related activity
  - Suspicious behavior

1. Normal Use:

```
vol.exe -f cridex.vmem --profile=WinXPSP3x86 wndscan
```

2. Pipe Output to a Text File:

```
vol.exe -f cridex.vmem --profile=WinXPSP3x86 wndscan > wnd.txt
```

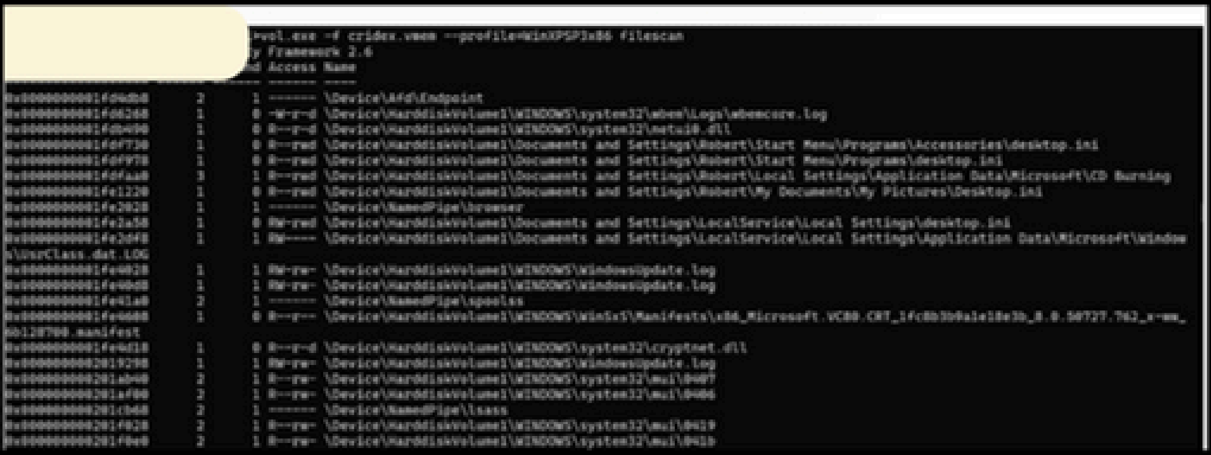


```
C:\Users[redacted]Downloads\vol>vol.exe -f cridex.vmem --profile=WinXPSP3x86 wndscan
Volatility Foundation Volatility Framework 2.6
**************************************************
WindowStation: 0x201e328, Name: Service-0x0-3e5$, Next: 0x8224Bfa0
SessionId: 0, AtomTable: 0xe19aa008, Interactive: False
Desktops: Default
ptiDrawingClipboard: pid - tid -
spwndClipOpen: 0x0, spwndClipViewer: 0x0
cNumClipFormats: 0, iClipSerialNumber: 0
pClipBase: 0x0, Formats:
**************************************************
WindowStation: 0x2448fa0, Name: SAWinSta, Next: 0x0
SessionId: 0, AtomTable: 0xe18089a0, Interactive: False
Desktops: SADesktop
ptiDrawingClipboard: pid - tid -
spwndClipOpen: 0x0, spwndClipViewer: 0x0
cNumClipFormats: 0, iClipSerialNumber: 0
pClipBase: 0x0, Formats:
**************************************************
WindowStation: 0x2029d50, Name: Service-0x0-3e4$, Next: 0x81e1e328
SessionId: 0, AtomTable: 0xe17dc008, Interactive: False
Desktops: Default
ptiDrawingClipboard: pid - tid -
spwndClipOpen: 0x0, spwndClipViewer: 0x0
cNumClipFormats: 0, iClipSerialNumber: 0
pClipBase: 0x0, Formats:
**************************************************
WindowStation: 0x225a2a0, Name: WinSta0, Next: 0x821b8560
SessionId: 0, AtomTable: 0xe1759420, Interactive: True
```

Checking for Files: `filescan` Plugin

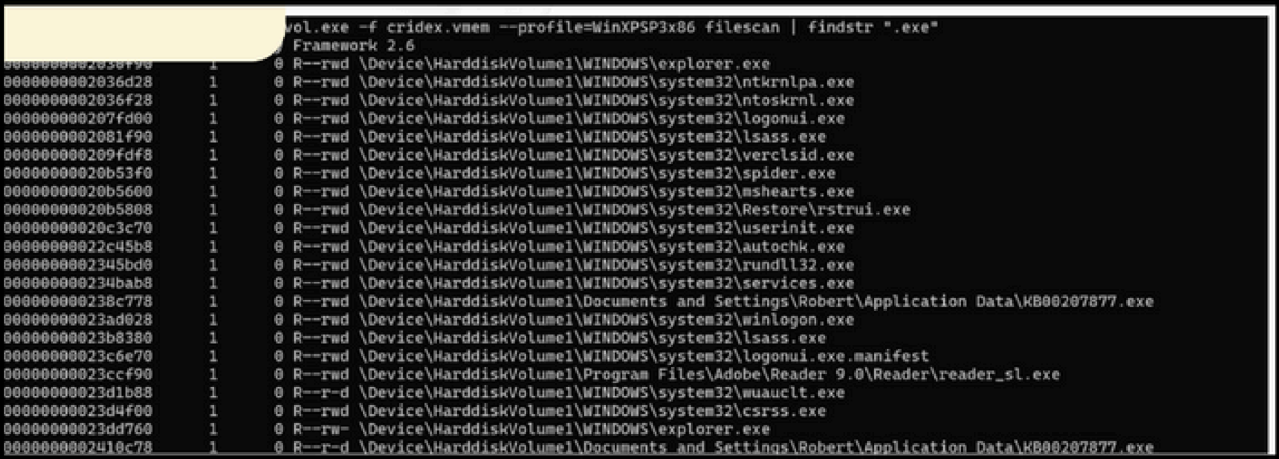
- The `filescan` plugin in Volatility is used to identify file objects in memory that may not have been mapped to disk.
- This is particularly useful for detecting hidden or injected files that could be used by malware.

```
vol.exe -f cridex.vmem --profile=WinXPSP3x86 filescan
```



```
[redacted]>vol.exe -f cridex.vmem --profile=WinXPSP3x86 filescan
y Framework 2.6
nd Access Name
------ ------ ----
0x0000000001fd4db8      2      1 ------ \Device\Afd\Endpoint
0x0000000001fd6268      1      0 -W-r-d \Device\HarddiskVolume1\WINDOWS\system32\wbem\Logs\wbemcore.log
0x0000000001fdb490      1      0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\netui0.dll
0x0000000001fdf730      1      0 R--rwd \Device\HarddiskVolume1\Documents and Settings\Robert\Start Menu\Programs\Accessories\desktop.ini
0x0000000001fdf978      1      0 R--rwd \Device\HarddiskVolume1\Documents and Settings\Robert\Start Menu\Programs\desktop.ini
0x0000000001fdfaa8      3      1 R--rwd \Device\HarddiskVolume1\Documents and Settings\Robert\Local Settings\Application Data\Microsoft\CD Burning
0x0000000001fe1220      1      0 R--rwd \Device\HarddiskVolume1\Documents and Settings\Robert\My Documents\My Pictures\Desktop.ini
0x0000000001fe2028      1      1 ------ \Device\NamedPipe\browser
0x0000000001fe2a58      1      0 RW-rwd \Device\HarddiskVolume1\Documents and Settings\LocalService\Local Settings\desktop.ini
0x0000000001fe2df8      1      1 RW---- \Device\HarddiskVolume1\Documents and Settings\LocalService\Local Settings\Application Data\Microsoft\Window
s\UsrClass.dat.LOG
0x0000000001fe4028      1      1 RW-rw- \Device\HarddiskVolume1\WINDOWS\WindowsUpdate.log
0x0000000001fe40d8      1      1 RW-rw- \Device\HarddiskVolume1\WINDOWS\WindowsUpdate.log
0x0000000001fe41a0      2      1 ------ \Device\NamedPipe\spoolss
0x0000000001fe4608      1      0 R--r-- \Device\HarddiskVolume1\WINDOWS\WinSxS\Manifests\x86_Microsoft.VC80.CRT_1fc8b3b9a1e18e3b_8.0.50727.762_x-ww_
6b128700.manifest
0x0000000001fe4d18      1      0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\cryptnet.dll
0x0000000002019298      1      1 RW-rw- \Device\HarddiskVolume1\WINDOWS\WindowsUpdate.log
0x000000000201ab40      2      1 R--rw- \Device\HarddiskVolume1\WINDOWS\system32\mui\0407
0x000000000201af80      2      1 R--rw- \Device\HarddiskVolume1\WINDOWS\system32\mui\0406
0x000000000201cb68      2      1 ------ \Device\NamedPipe\lsass
0x000000000201f028      2      1 R--rw- \Device\HarddiskVolume1\WINDOWS\system32\mui\0419
0x000000000201f0e0      2      1 R--rw- \Device\HarddiskVolume1\WINDOWS\system32\mui\041b
```

Note : regarding to your time . we can use | findstr in windows or | grep in Linux to search for specific file on this

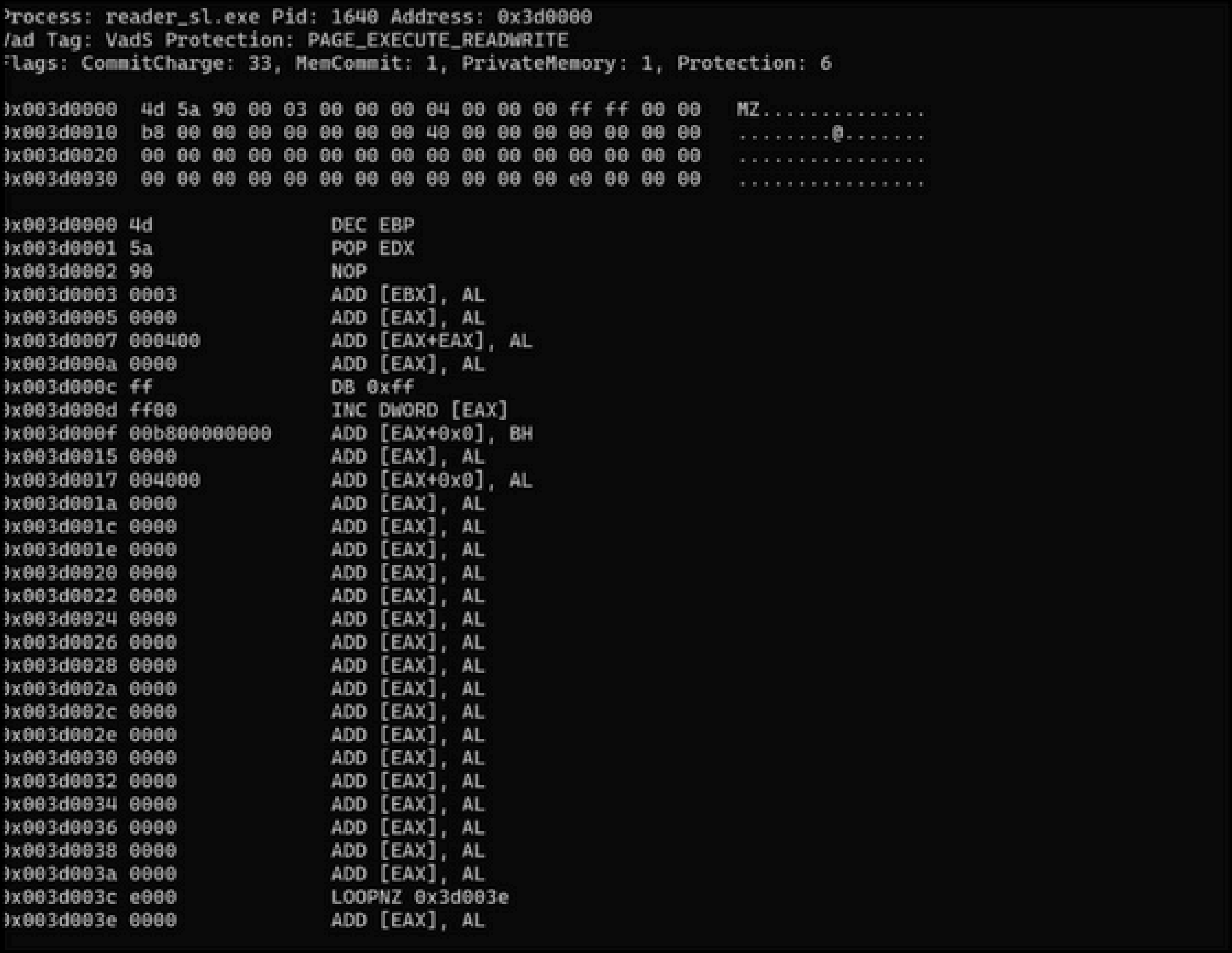


```
[redacted]vol.exe -f cridex.vmem --profile=WinXPSP3x86 filescan | findstr ".exe"
          Framework 2.6
                             0 R--rwd \Device\HarddiskVolume1\WINDOWS\explorer.exe
0000000002036d28       1     0 R--rwd \Device\HarddiskVolume1\WINDOWS\system32\ntkrnlpa.exe
0000000002036f28       1     0 R--rwd \Device\HarddiskVolume1\WINDOWS\system32\ntoskrnl.exe
000000000207fd00       1     0 R--rwd \Device\HarddiskVolume1\WINDOWS\system32\logonui.exe
0000000002081f90       1     0 R--rwd \Device\HarddiskVolume1\WINDOWS\system32\lsass.exe
000000000209fdf8       1     0 R--rwd \Device\HarddiskVolume1\WINDOWS\system32\verclsid.exe
00000000020b53f0       1     0 R--rwd \Device\HarddiskVolume1\WINDOWS\system32\spider.exe
00000000020b5600       1     0 R--rwd \Device\HarddiskVolume1\WINDOWS\system32\mshearts.exe
00000000020b5808       1     0 R--rwd \Device\HarddiskVolume1\WINDOWS\system32\Restore\rstrui.exe
00000000020c3c70       1     0 R--rwd \Device\HarddiskVolume1\WINDOWS\system32\userinit.exe
00000000022c45b8       1     0 R--rwd \Device\HarddiskVolume1\WINDOWS\system32\autochk.exe
00000000002345bd0      1     0 R--rwd \Device\HarddiskVolume1\WINDOWS\system32\rundll32.exe
000000000234bab8       1     0 R--rwd \Device\HarddiskVolume1\WINDOWS\system32\services.exe
000000000238c778       1     0 R--rwd \Device\HarddiskVolume1\Documents and Settings\Robert\Application Data\KB00207877.exe
00000000023ad028       1     0 R--rwd \Device\HarddiskVolume1\WINDOWS\system32\winlogon.exe
00000000023b8380       1     0 R--rwd \Device\HarddiskVolume1\WINDOWS\system32\lsass.exe
00000000023c6e70       1     0 R--rwd \Device\HarddiskVolume1\WINDOWS\system32\logonui.exe.manifest
00000000023ccf90       1     0 R--rwd \Device\HarddiskVolume1\Program Files\Adobe\Reader 9.0\Reader\reader_sl.exe
00000000023d1b88       1     0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\wuauclt.exe
00000000023d4f00       1     0 R--rwd \Device\HarddiskVolume1\WINDOWS\system32\csrss.exe
00000000023dd760       1     0 R--rw- \Device\HarddiskVolume1\WINDOWS\explorer.exe
0000000002410c78       1     0 R--r-d \Device\HarddiskVolume1\Documents and Settings\Robert\Application Data\KB00207877.exe
```

Checking for Malware: `malfind` Plugin

- The `malfind` plugin in Volatility is a powerful tool for identifying **potential malware** within a memory dump.
- It scans for **injected code** or **anomalous memory sections** that are typically associated with malware.

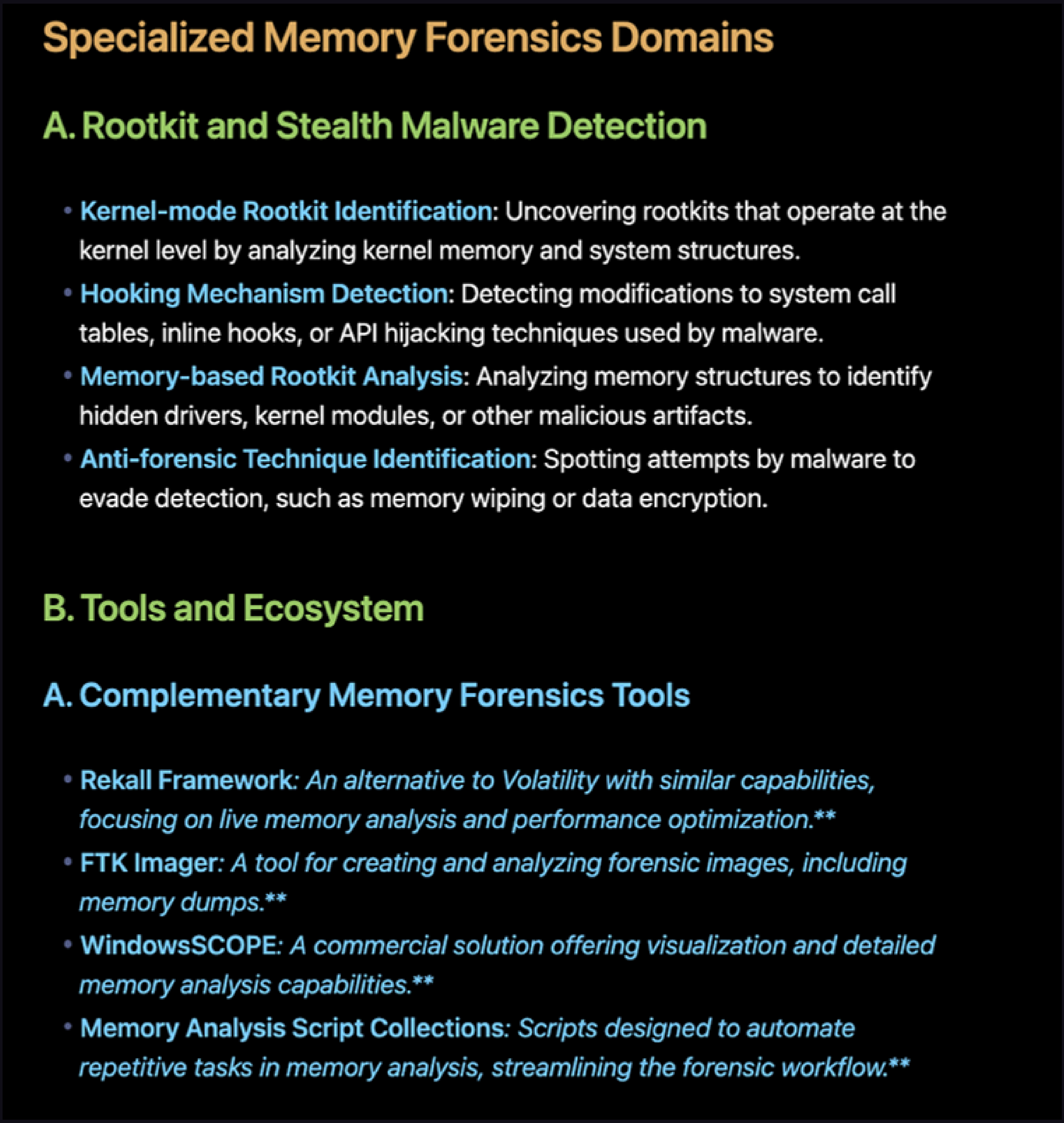
```
vol.exe -f cridex.vmem --profile=WinXPSP3x86 malfind
```



```
Process: reader_sl.exe Pid: 1640 Address: 0x3d0000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 33, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x003d0000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x003d0010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x003d0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x003d0030  00 00 00 00 00 00 00 00 00 00 00 00 e0 00 00 00   ................

0x003d0000 4d                DEC EBP
0x003d0001 5a                POP EDX
0x003d0002 90                NOP
0x003d0003 0003              ADD [EBX], AL
0x003d0005 0000              ADD [EAX], AL
0x003d0007 000400            ADD [EAX+EAX], AL
0x003d000a 0000              ADD [EAX], AL
0x003d000c ff                DB 0xff
0x003d000d ff00              INC DWORD [EAX]
0x003d000f 00b800000000      ADD [EAX+0x0], BH
0x003d0015 0000              ADD [EAX], AL
0x003d0017 004000            ADD [EAX+0x0], AL
0x003d001a 0000              ADD [EAX], AL
0x003d001c 0000              ADD [EAX], AL
0x003d001e 0000              ADD [EAX], AL
0x003d0020 0000              ADD [EAX], AL
0x003d0022 0000              ADD [EAX], AL
0x003d0024 0000              ADD [EAX], AL
0x003d0026 0000              ADD [EAX], AL
0x003d0028 0000              ADD [EAX], AL
0x003d002a 0000              ADD [EAX], AL
0x003d002c 0000              ADD [EAX], AL
0x003d002e 0000              ADD [EAX], AL
0x003d0030 0000              ADD [EAX], AL
0x003d0032 0000              ADD [EAX], AL
0x003d0034 0000              ADD [EAX], AL
0x003d0036 0000              ADD [EAX], AL
0x003d0038 0000              ADD [EAX], AL
0x003d003a 0000              ADD [EAX], AL
0x003d003c e000              LOOPNZ 0x3d003e
0x003d003e 0000              ADD [EAX], AL
```

C. Memory Artifact Reconstruction

- **Registry Hive Recovery**: Extracting and analyzing registry hives from memory to uncover configuration changes or malware persistence mechanisms.
- **Network Connection Tracking**: Identifying live or historical network connections to analyze potential data exfiltration or communication with command-and-control (C2) servers.
- **Authentication Session Forensics**: Investigating authentication tokens, session IDs, and user credential usage stored in memory.
- **Cached Credentials Examination**: Analyzing cached credentials to detect potential credential harvesting or misuse.



Specialized Memory Forensics Domains

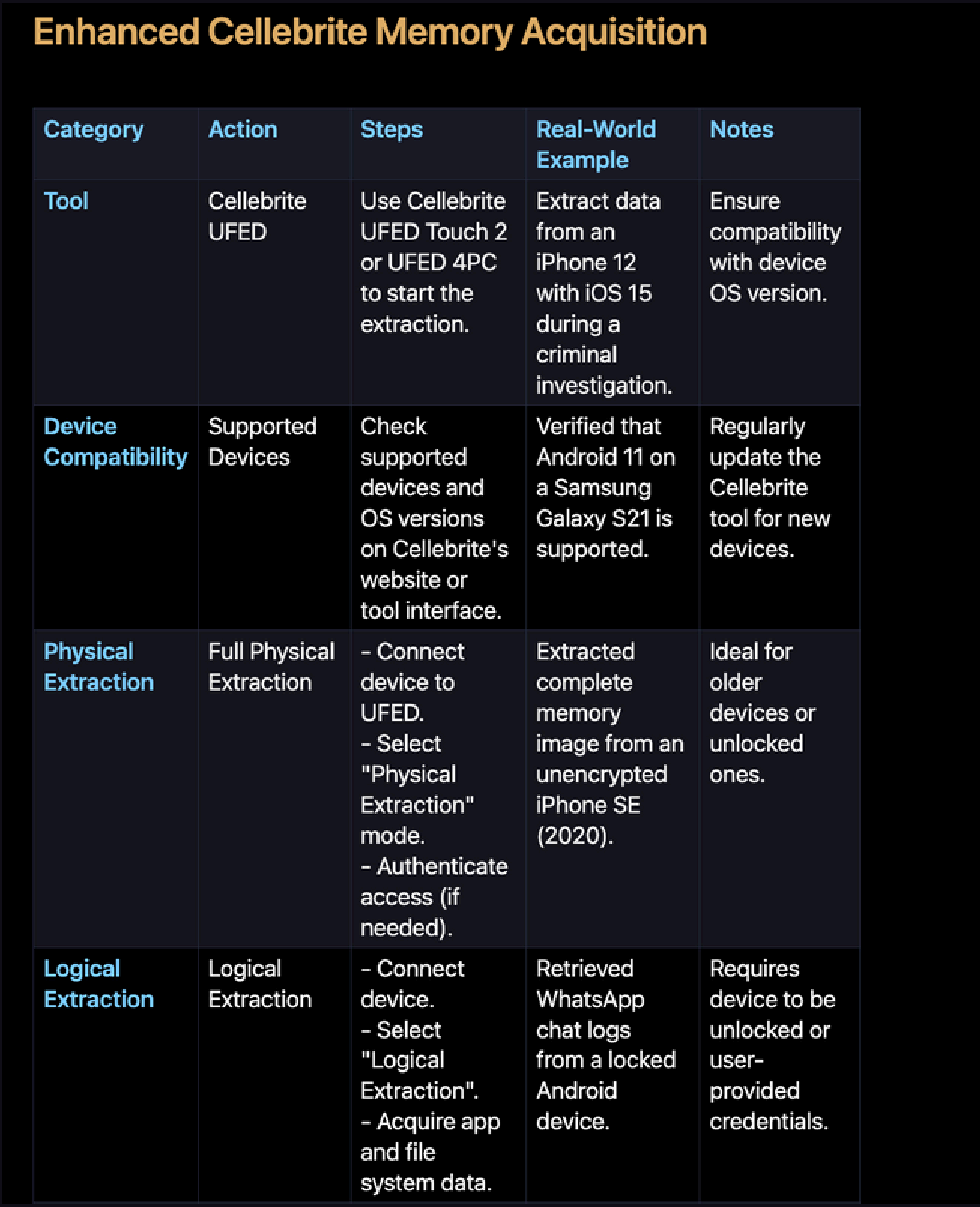
A. Rootkit and Stealth Malware Detection

- **Kernel-mode Rootkit Identification:** Uncovering rootkits that operate at the kernel level by analyzing kernel memory and system structures.
- **Hooking Mechanism Detection:** Detecting modifications to system call tables, inline hooks, or API hijacking techniques used by malware.
- **Memory-based Rootkit Analysis:** Analyzing memory structures to identify hidden drivers, kernel modules, or other malicious artifacts.
- **Anti-forensic Technique Identification:** Spotting attempts by malware to evade detection, such as memory wiping or data encryption.

B. Tools and Ecosystem

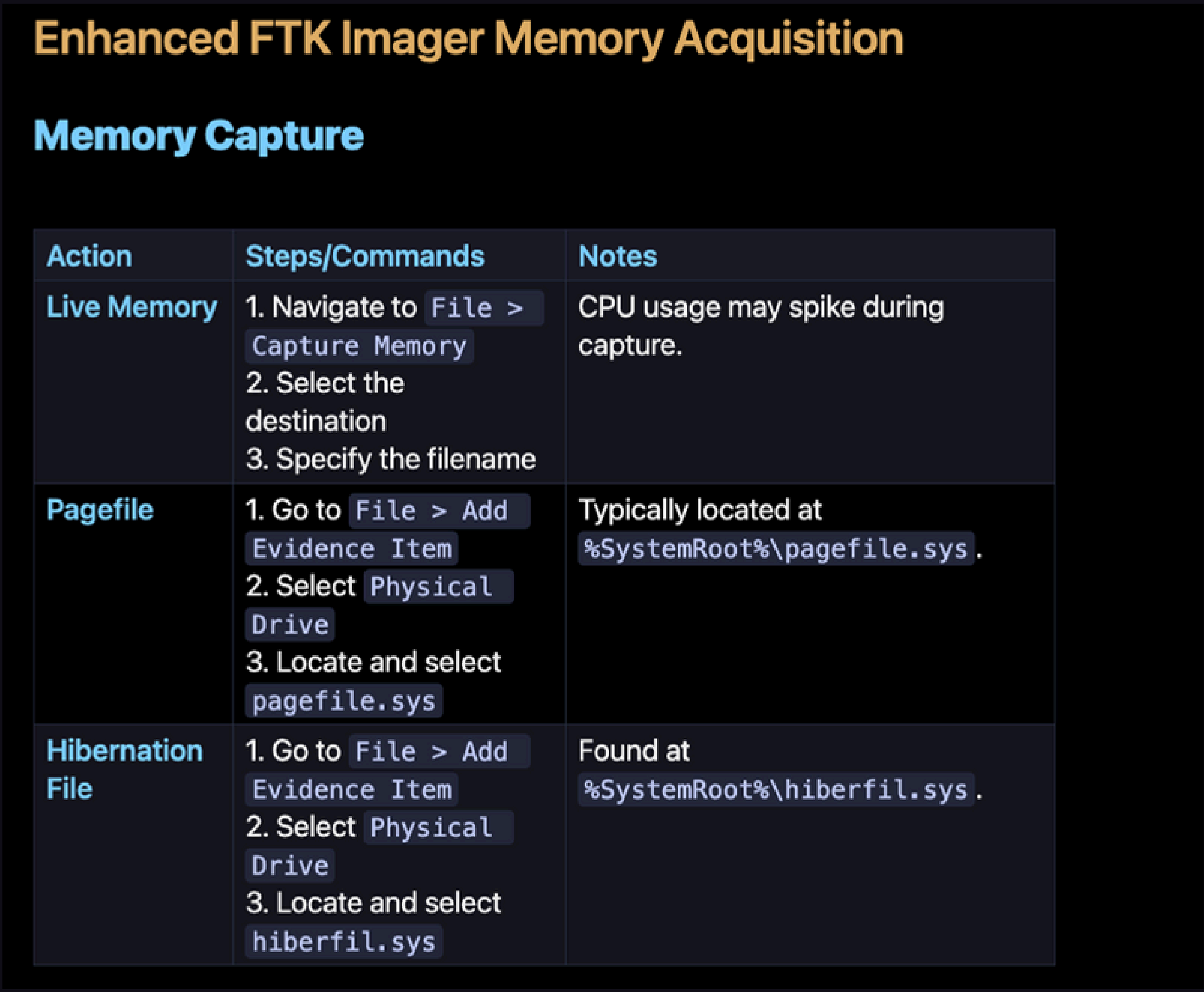
A. Complementary Memory Forensics Tools

- **Rekall Framework**: *An alternative to Volatility with similar capabilities, focusing on live memory analysis and performance optimization.***
- **FTK Imager**: *A tool for creating and analyzing forensic images, including memory dumps.***
- **WindowsSCOPE**: *A commercial solution offering visualization and detailed memory analysis capabilities.***
- **Memory Analysis Script Collections**: *Scripts designed to automate repetitive tasks in memory analysis, streamlining the forensic workflow.***



Enhanced Cellebrite Memory Acquisition

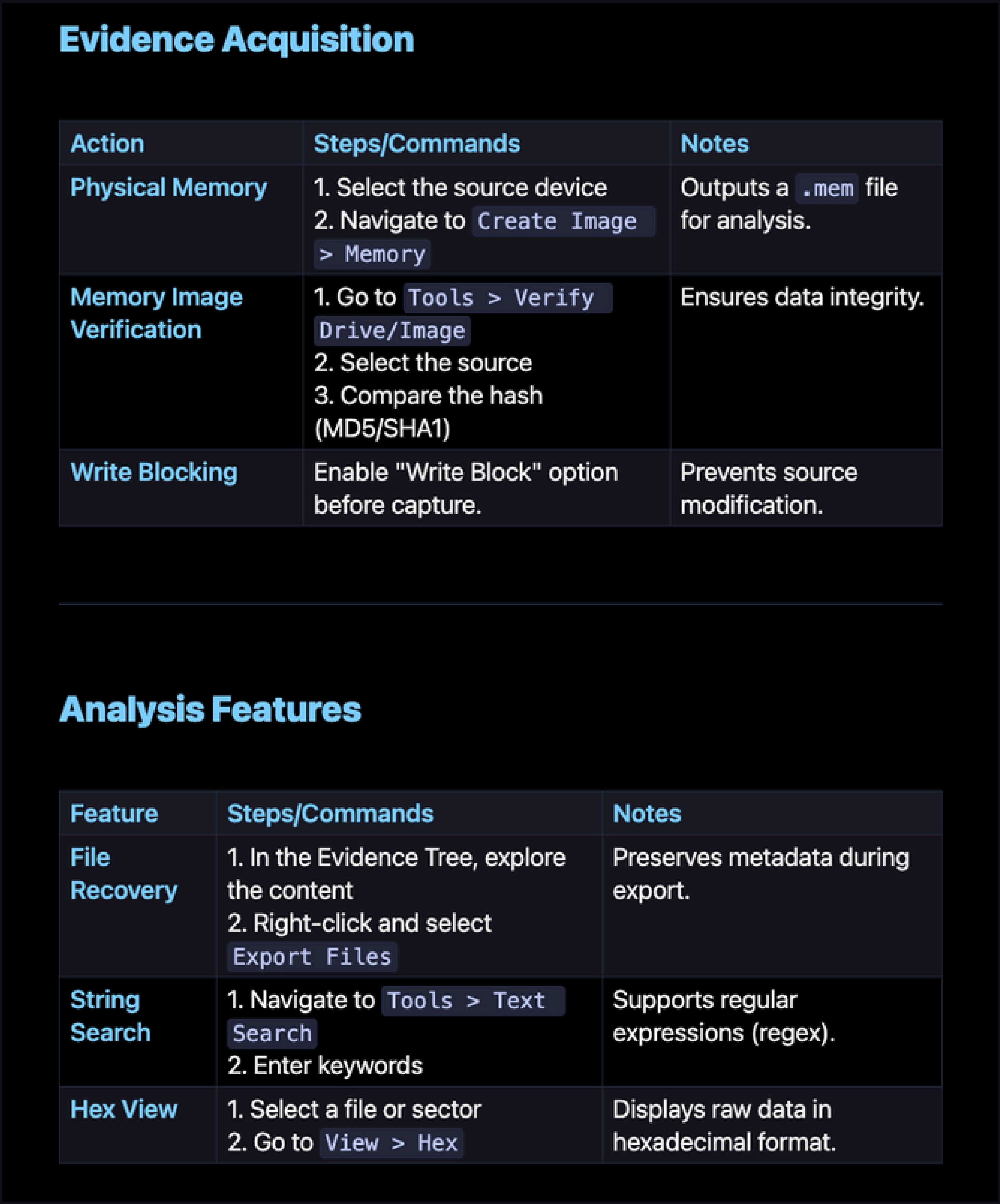
Category	Action	Steps	Real-World Example	Notes
Tool	Cellebrite UFED	Use Cellebrite UFED Touch 2 or UFED 4PC to start the extraction.	Extract data from an iPhone 12 with iOS 15 during a criminal investigation.	Ensure compatibility with device OS version.
Device Compatibility	Supported Devices	Check supported devices and OS versions on Cellebrite's website or tool interface.	Verified that Android 11 on a Samsung Galaxy S21 is supported.	Regularly update the Cellebrite tool for new devices.
Physical Extraction	Full Physical Extraction	- Connect device to UFED. - Select "Physical Extraction" mode. - Authenticate access (if needed).	Extracted complete memory image from an unencrypted iPhone SE (2020).	Ideal for older devices or unlocked ones.
Logical Extraction	Logical Extraction	- Connect device. - Select "Logical Extraction". - Acquire app and file system data.	Retrieved WhatsApp chat logs from a locked Android device.	Requires device to be unlocked or user-provided credentials.



Enhanced FTK Imager Memory Acquisition

Memory Capture

Action	Steps/Commands	Notes
Live Memory	1. Navigate to `File > Capture Memory` 2. Select the destination 3. Specify the filename	CPU usage may spike during capture.
Pagefile	1. Go to `File > Add Evidence Item` 2. Select `Physical Drive` 3. Locate and select `pagefile.sys`	Typically located at `%SystemRoot%\pagefile.sys`.
Hibernation File	1. Go to `File > Add Evidence Item` 2. Select `Physical Drive` 3. Locate and select `hiberfil.sys`	Found at `%SystemRoot%\hiberfil.sys`.

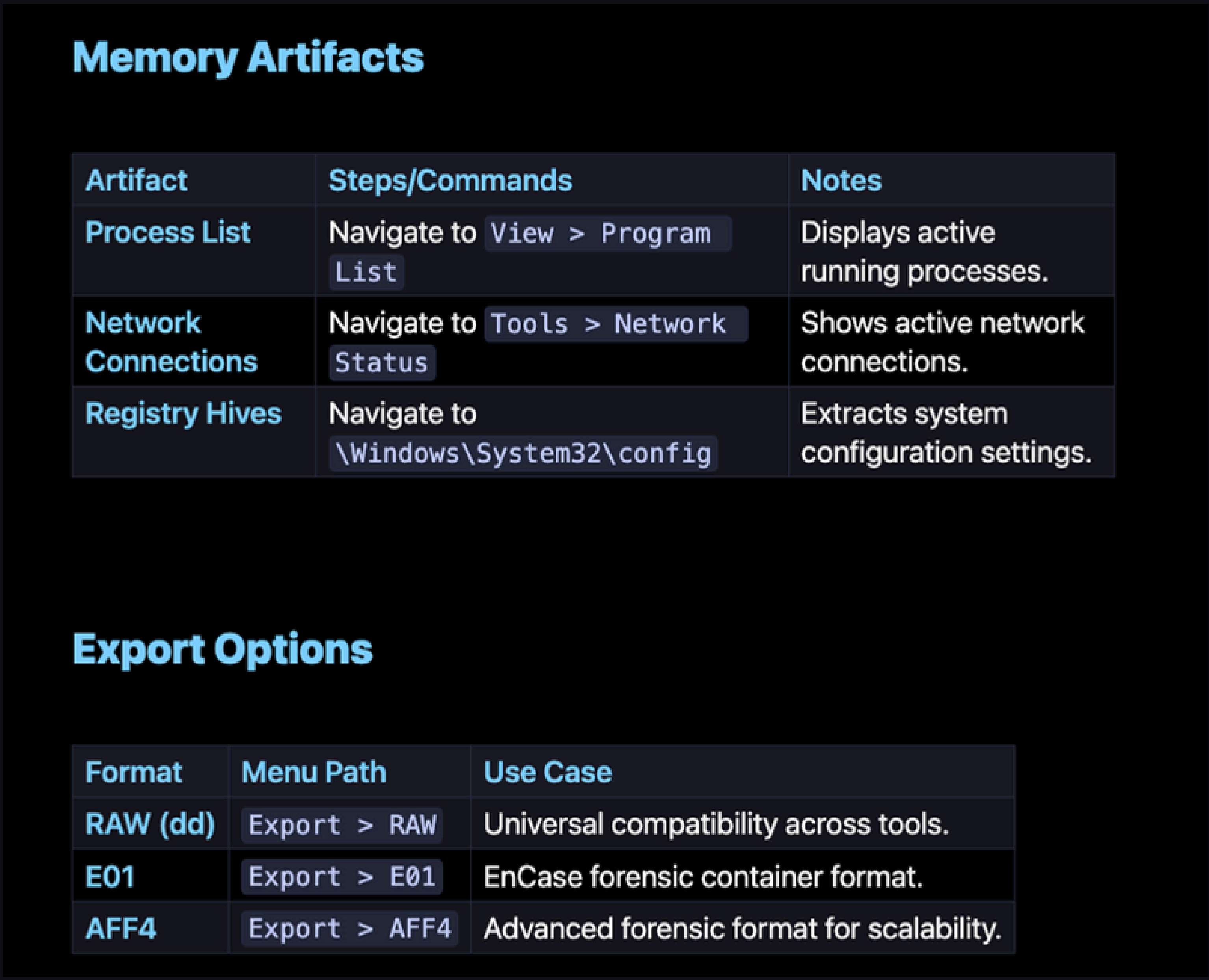


Evidence Acquisition

Action	Steps/Commands	Notes
Physical Memory	1. Select the source device 2. Navigate to `Create Image > Memory`	Outputs a `.mem` file for analysis.
Memory Image Verification	1. Go to `Tools > Verify Drive/Image` 2. Select the source 3. Compare the hash (MD5/SHA1)	Ensures data integrity.
Write Blocking	Enable "Write Block" option before capture.	Prevents source modification.

Analysis Features

Feature	Steps/Commands	Notes
File Recovery	1. In the Evidence Tree, explore the content 2. Right-click and select `Export Files`	Preserves metadata during export.
String Search	1. Navigate to `Tools > Text Search` 2. Enter keywords	Supports regular expressions (regex).
Hex View	1. Select a file or sector 2. Go to `View > Hex`	Displays raw data in hexadecimal format.

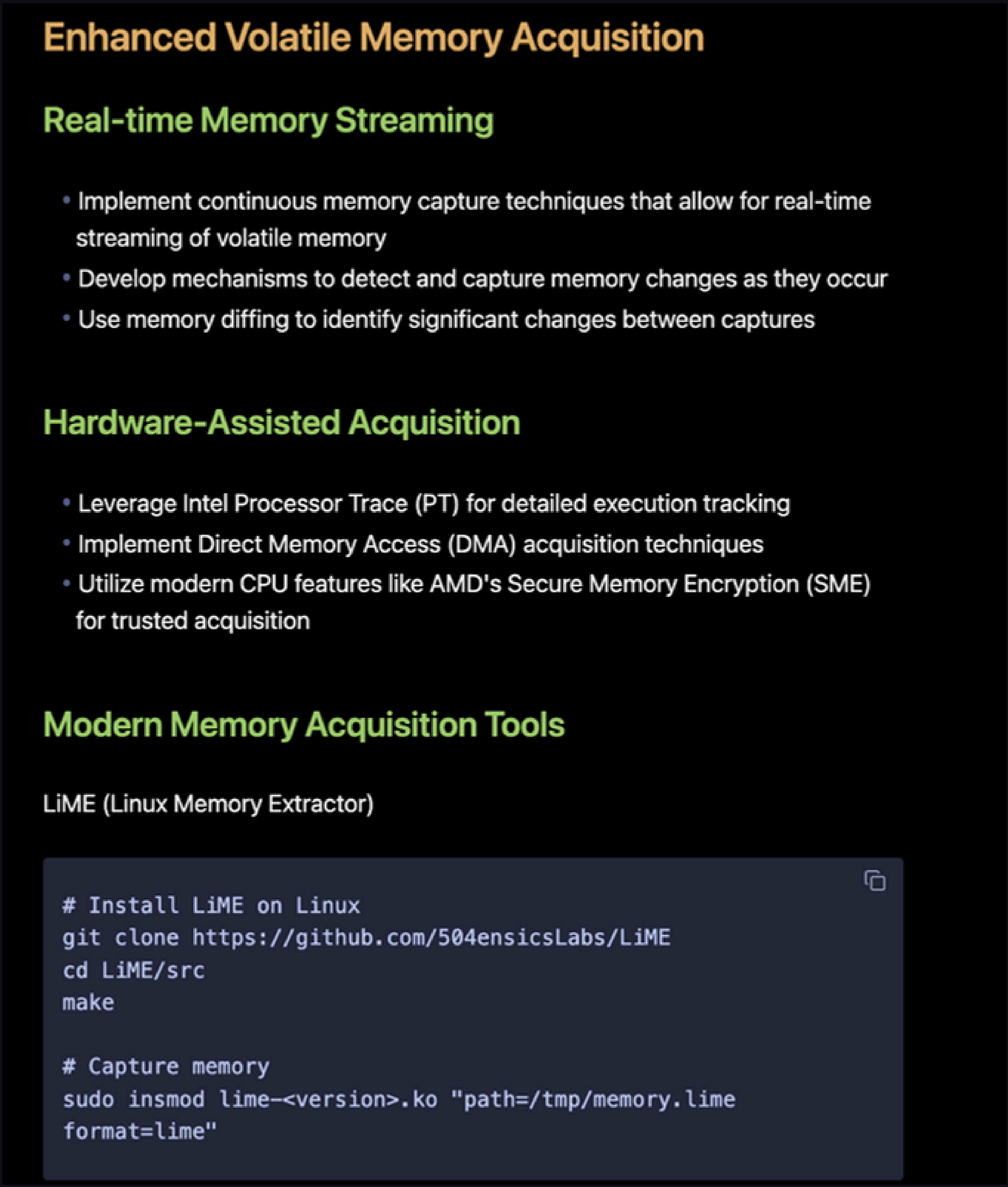


Memory Artifacts

Artifact	Steps/Commands	Notes
Process List	Navigate to `View > Program List`	Displays active running processes.
Network Connections	Navigate to `Tools > Network Status`	Shows active network connections.
Registry Hives	Navigate to `\Windows\System32\config`	Extracts system configuration settings.

Export Options

Format	Menu Path	Use Case
RAW (dd)	`Export > RAW`	Universal compatibility across tools.
E01	`Export > E01`	EnCase forensic container format.
AFF4	`Export > AFF4`	Advanced forensic format for scalability.



Enhanced Volatile Memory Acquisition

Real-time Memory Streaming

- Implement continuous memory capture techniques that allow for real-time streaming of volatile memory
- Develop mechanisms to detect and capture memory changes as they occur
- Use memory diffing to identify significant changes between captures

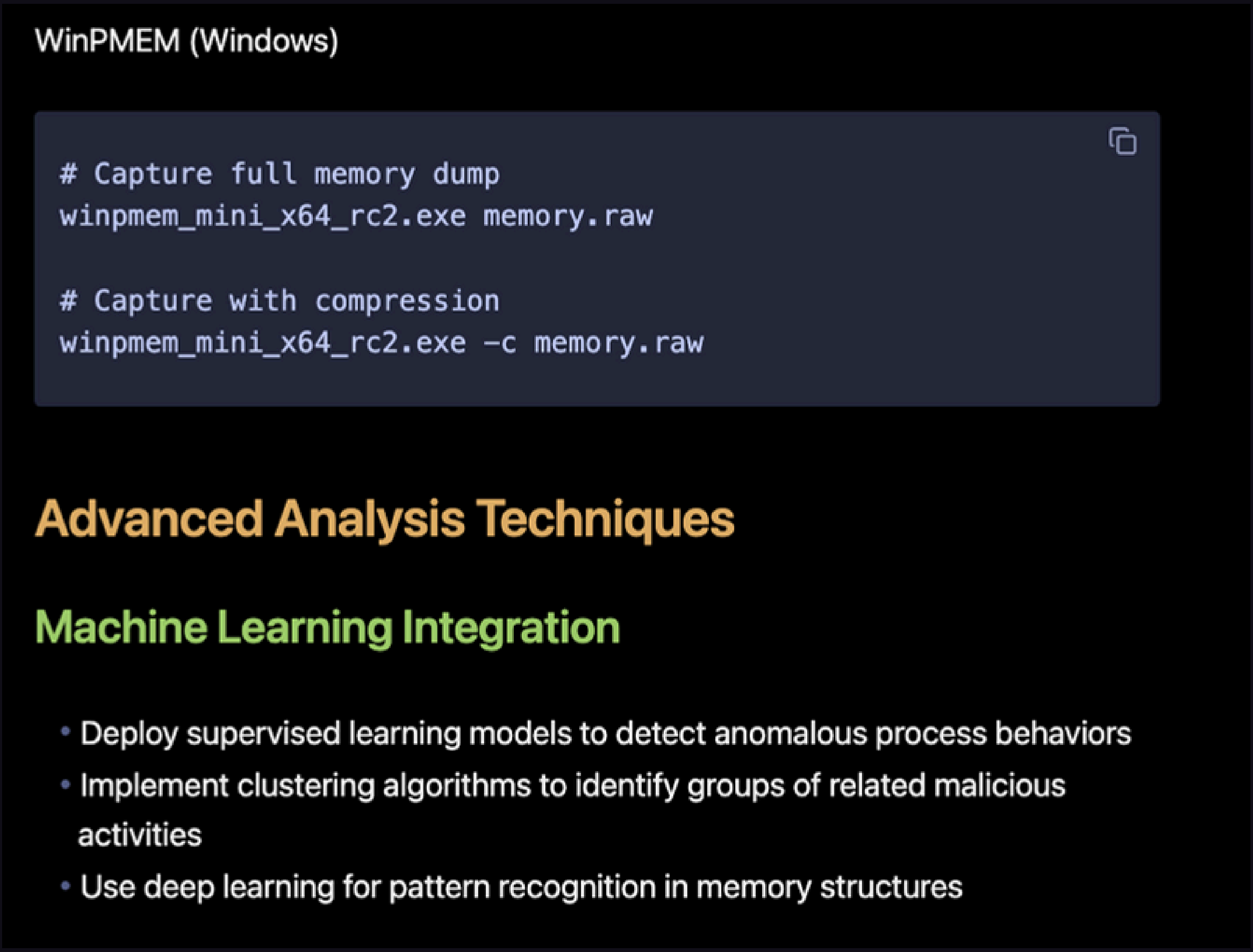
Hardware-Assisted Acquisition

- Leverage Intel Processor Trace (PT) for detailed execution tracking
- Implement Direct Memory Access (DMA) acquisition techniques
- Utilize modern CPU features like AMD's Secure Memory Encryption (SME) for trusted acquisition

Modern Memory Acquisition Tools

LiME (Linux Memory Extractor)

```
# Install LiME on Linux
git clone https://github.com/504ensicsLabs/LiME
cd LiME/src
make

# Capture memory
sudo insmod lime-<version>.ko "path=/tmp/memory.lime
format=lime"
```



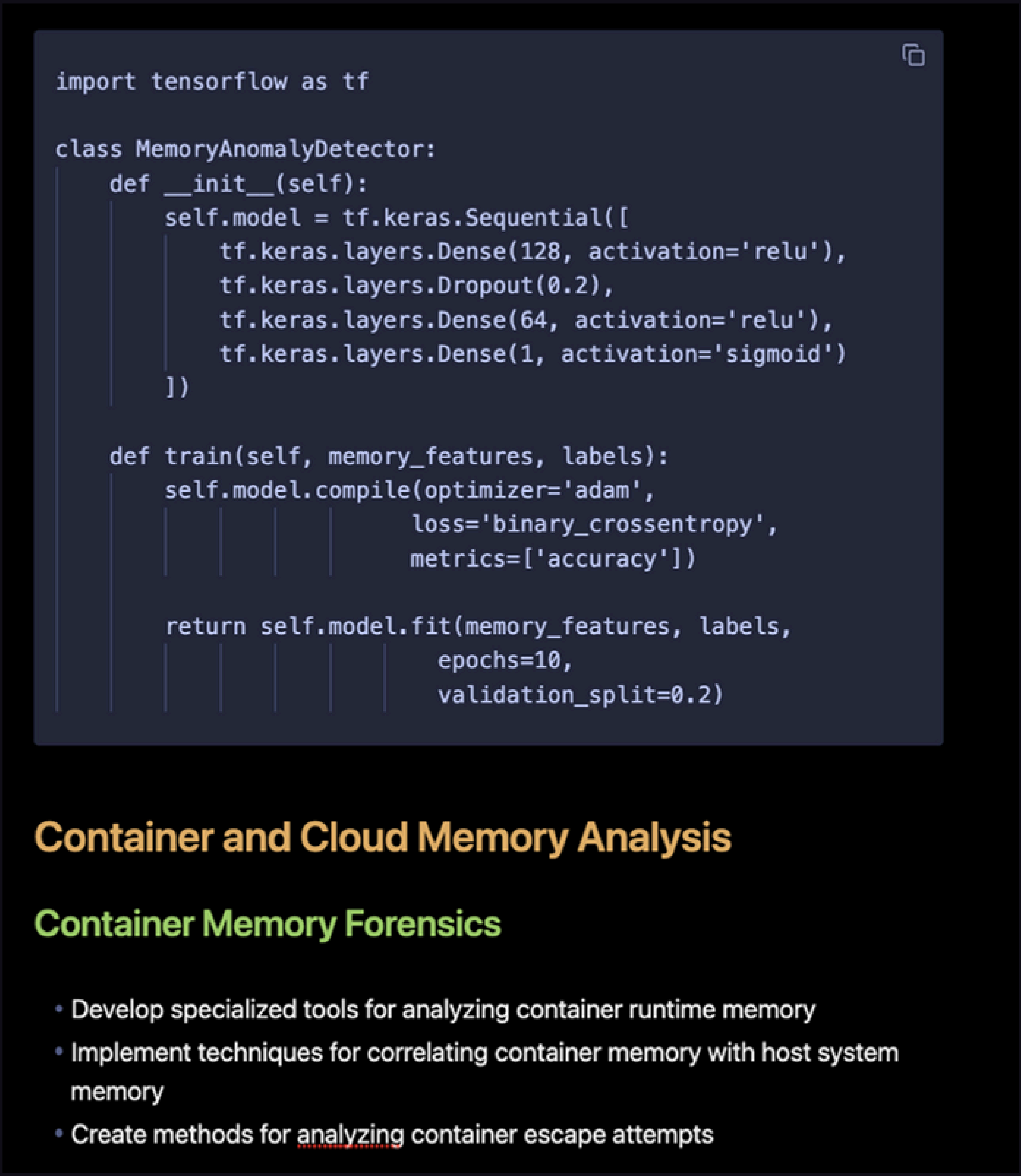
WinPMEM (Windows)

```
# Capture full memory dump
winpmem_mini_x64_rc2.exe memory.raw

# Capture with compression
winpmem_mini_x64_rc2.exe -c memory.raw
```

Advanced Analysis Techniques

Machine Learning Integration

- Deploy supervised learning models to detect anomalous process behaviors
- Implement clustering algorithms to identify groups of related malicious activities
- Use deep learning for pattern recognition in memory structures

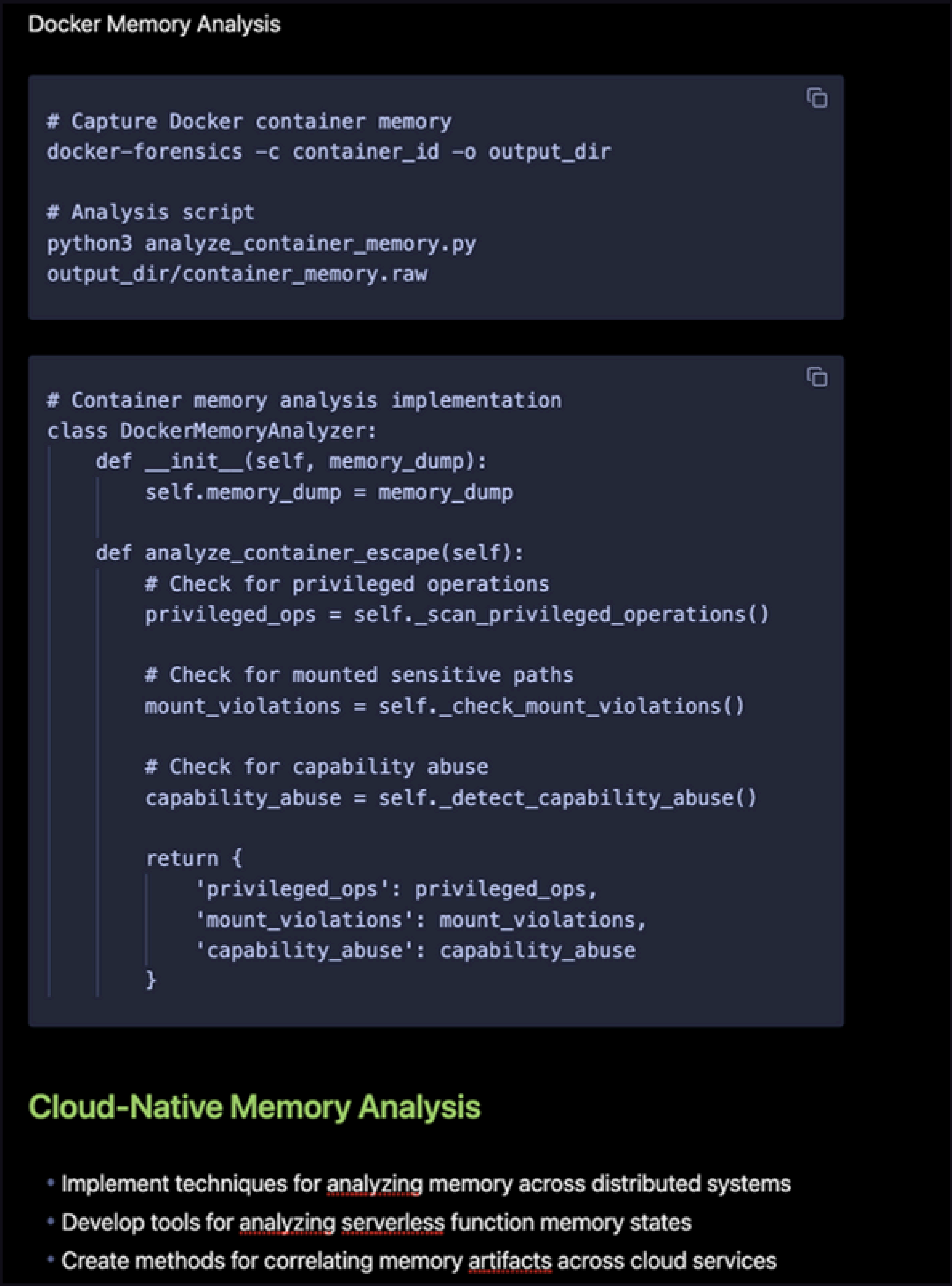


```
import tensorflow as tf

class MemoryAnomalyDetector:
    def __init__(self):
        self.model = tf.keras.Sequential([
            tf.keras.layers.Dense(128, activation='relu'),
            tf.keras.layers.Dropout(0.2),
            tf.keras.layers.Dense(64, activation='relu'),
            tf.keras.layers.Dense(1, activation='sigmoid')
        ])

    def train(self, memory_features, labels):
        self.model.compile(optimizer='adam',
                          loss='binary_crossentropy',
                          metrics=['accuracy'])

        return self.model.fit(memory_features, labels,
                            epochs=10,
                            validation_split=0.2)
```

Container and Cloud Memory Analysis

Container Memory Forensics

- Develop specialized tools for analyzing container runtime memory
- Implement techniques for correlating container memory with host system memory
- Create methods for analyzing container escape attempts



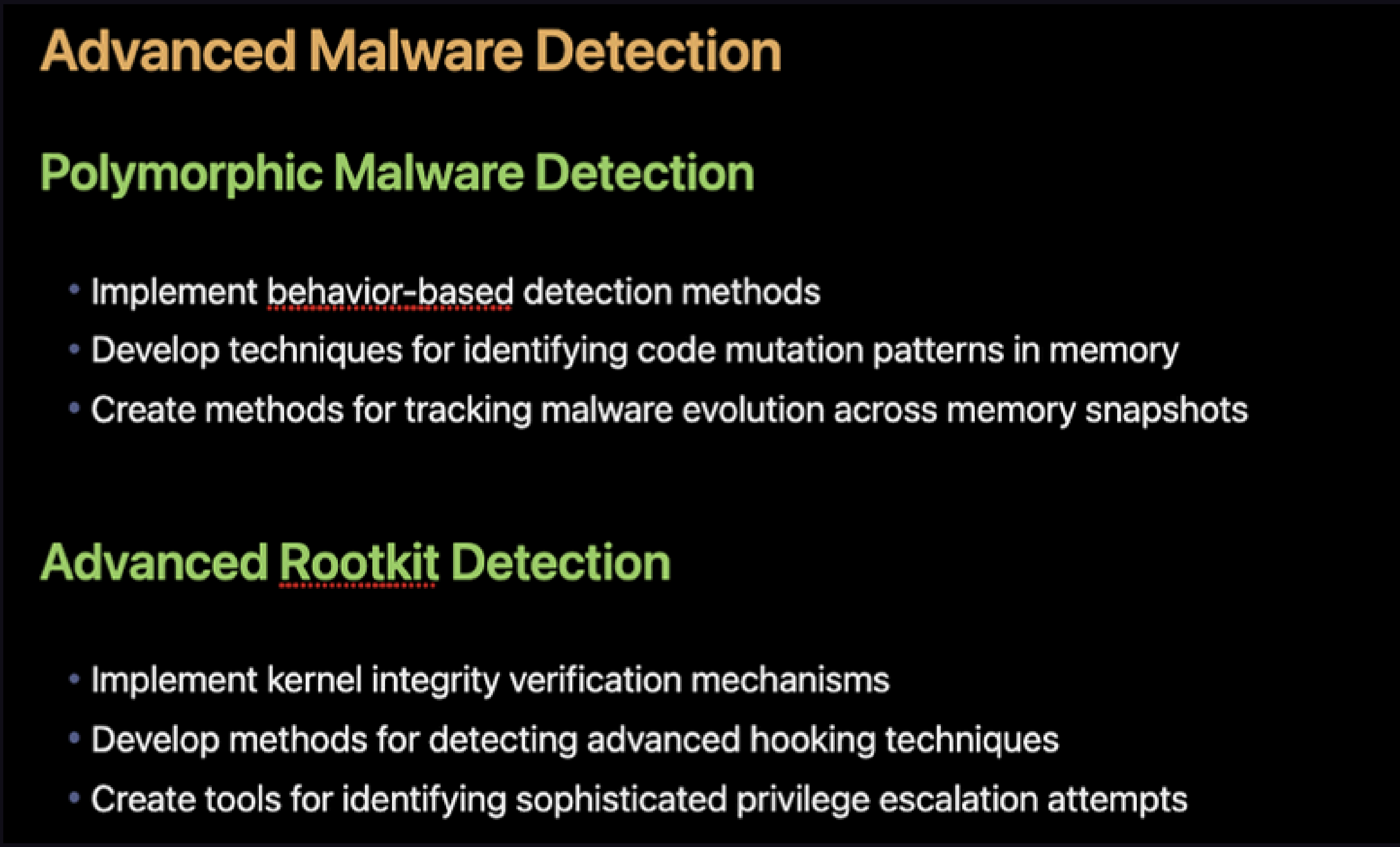
Docker Memory Analysis

```
# Capture Docker container memory
docker-forensics -c container_id -o output_dir

# Analysis script
python3 analyze_container_memory.py
output_dir/container_memory.raw
```

```
# Container memory analysis implementation
class DockerMemoryAnalyzer:
    def __init__(self, memory_dump):
        self.memory_dump = memory_dump

    def analyze_container_escape(self):
        # Check for privileged operations
        privileged_ops = self._scan_privileged_operations()

        # Check for mounted sensitive paths
        mount_violations = self._check_mount_violations()

        # Check for capability abuse
        capability_abuse = self._detect_capability_abuse()

        return {
            'privileged_ops': privileged_ops,
            'mount_violations': mount_violations,
            'capability_abuse': capability_abuse
        }
```

Cloud-Native Memory Analysis

- Implement techniques for analyzing memory across distributed systems
- Develop tools for analyzing serverless function memory states
- Create methods for correlating memory artifacts across cloud services



Advanced Malware Detection

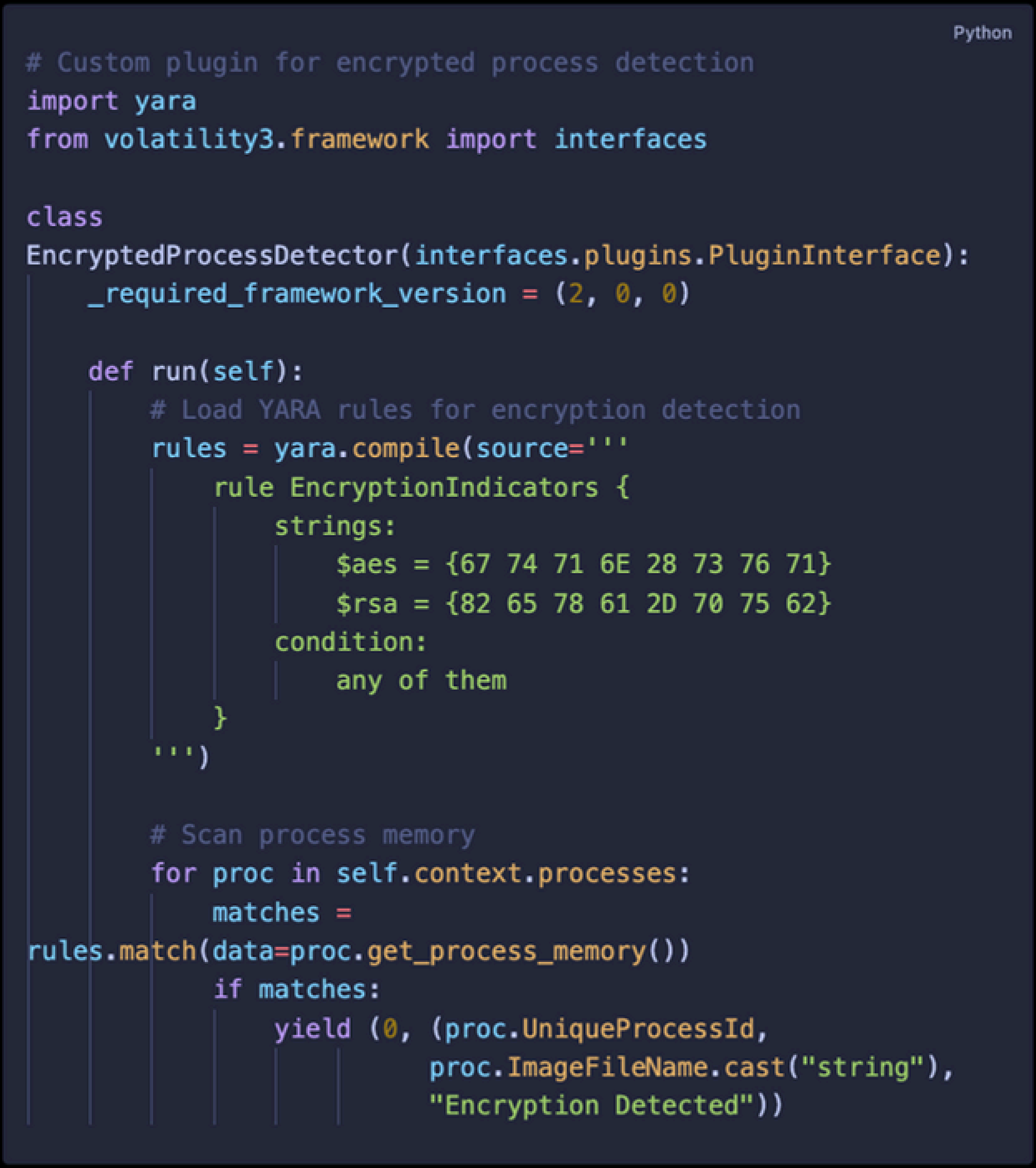
Polymorphic Malware Detection

- Implement behavior-based detection methods
- Develop techniques for identifying code mutation patterns in memory
- Create methods for tracking malware evolution across memory snapshots

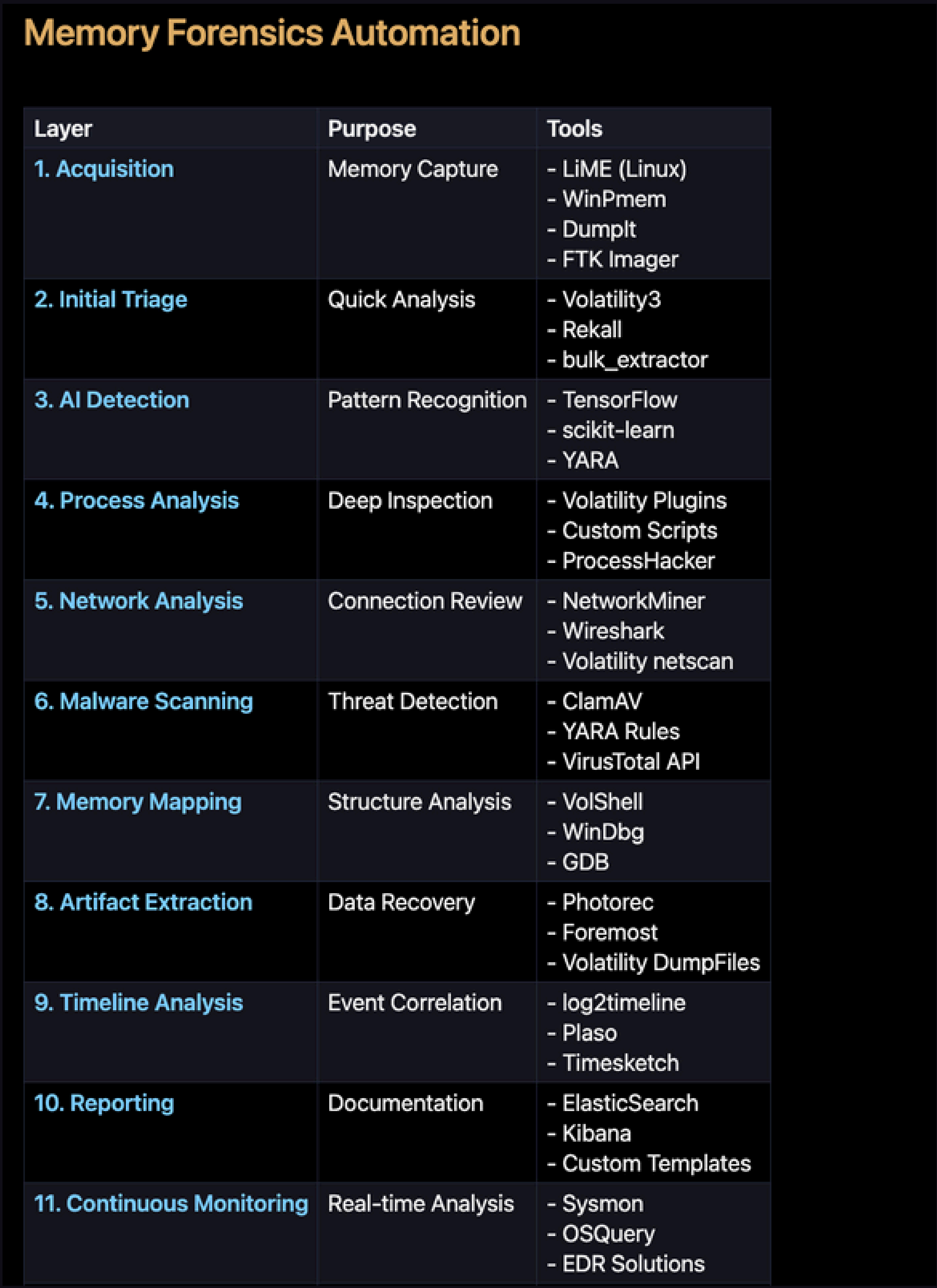
Advanced Rootkit Detection

- Implement kernel integrity verification mechanisms
- Develop methods for detecting advanced hooking techniques
- Create tools for identifying sophisticated privilege escalation attempts

Volatility 3 with Custom Plugins

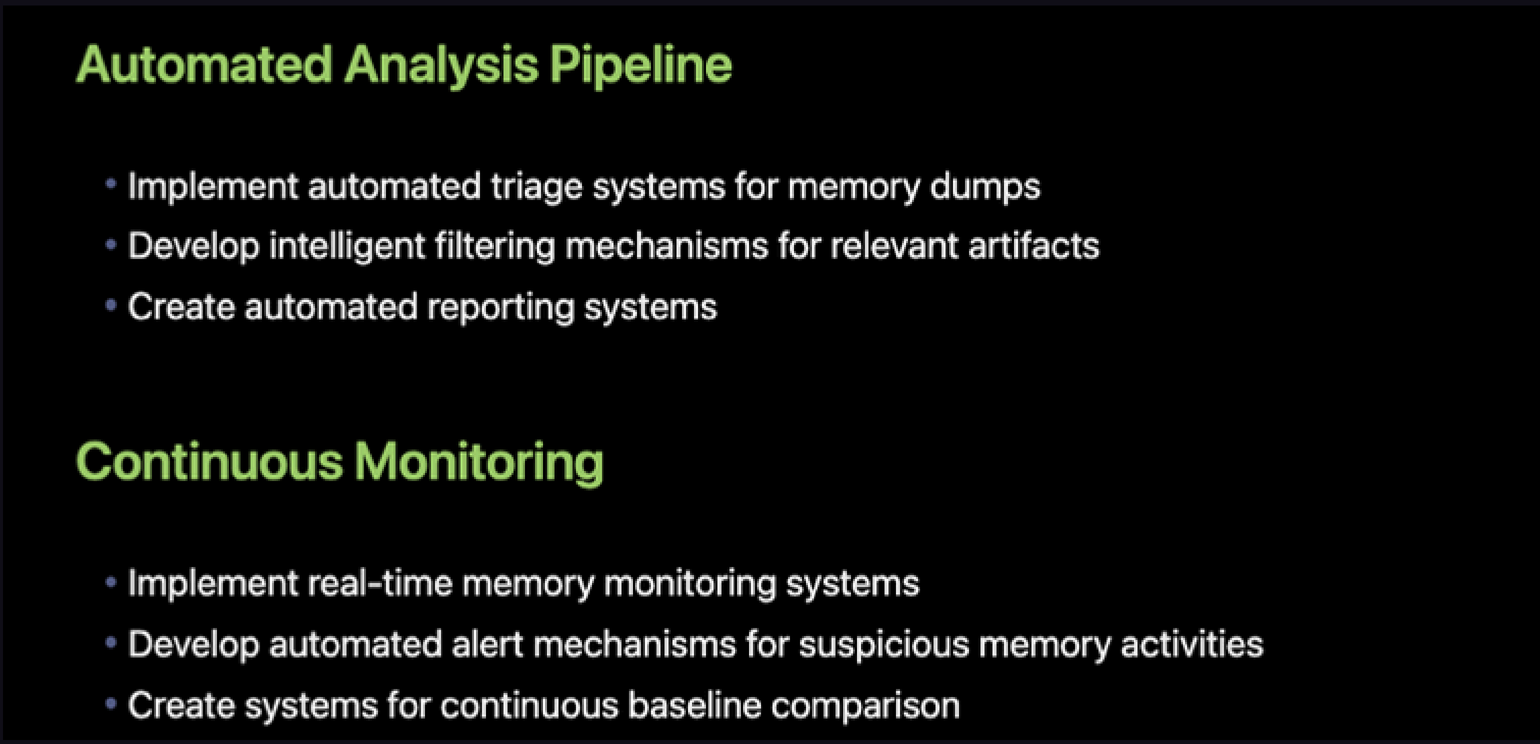


```
# Custom plugin for encrypted process detection
import yara
from volatility3.framework import interfaces

class
EncryptedProcessDetector(interfaces.plugins.PluginInterface):
    _required_framework_version = (2, 0, 0)

    def run(self):
        # Load YARA rules for encryption detection
        rules = yara.compile(source='''
            rule EncryptionIndicators {
                strings:
                    $aes = {67 74 71 6E 28 73 76 71}
                    $rsa = {82 65 78 61 2D 70 75 62}
                condition:
                    any of them
            }
        ''')

        # Scan process memory
        for proc in self.context.processes:
            matches =
rules.match(data=proc.get_process_memory())
            if matches:
                yield (0, (proc.UniqueProcessId,
                          proc.ImageFileName.cast("string"),
                          "Encryption Detected"))
```



Memory Forensics Automation

Layer	Purpose	Tools
1. Acquisition	Memory Capture	- LiME (Linux) - WinPmem - DumpIt - FTK Imager
2. Initial Triage	Quick Analysis	- Volatility3 - Rekall - bulk_extractor
3. AI Detection	Pattern Recognition	- TensorFlow - scikit-learn - YARA
4. Process Analysis	Deep Inspection	- Volatility Plugins - Custom Scripts - ProcessHacker
5. Network Analysis	Connection Review	- NetworkMiner - Wireshark - Volatility netscan
6. Malware Scanning	Threat Detection	- ClamAV - YARA Rules - VirusTotal API
7. Memory Mapping	Structure Analysis	- VolShell - WinDbg - GDB
8. Artifact Extraction	Data Recovery	- Photorec - Foremost - Volatility DumpFiles
9. Timeline Analysis	Event Correlation	- log2timeline - Plaso - Timesketch
10. Reporting	Documentation	- ElasticSearch - Kibana - Custom Templates
11. Continuous Monitoring	Real-time Analysis	- Sysmon - OSQuery - EDR Solutions

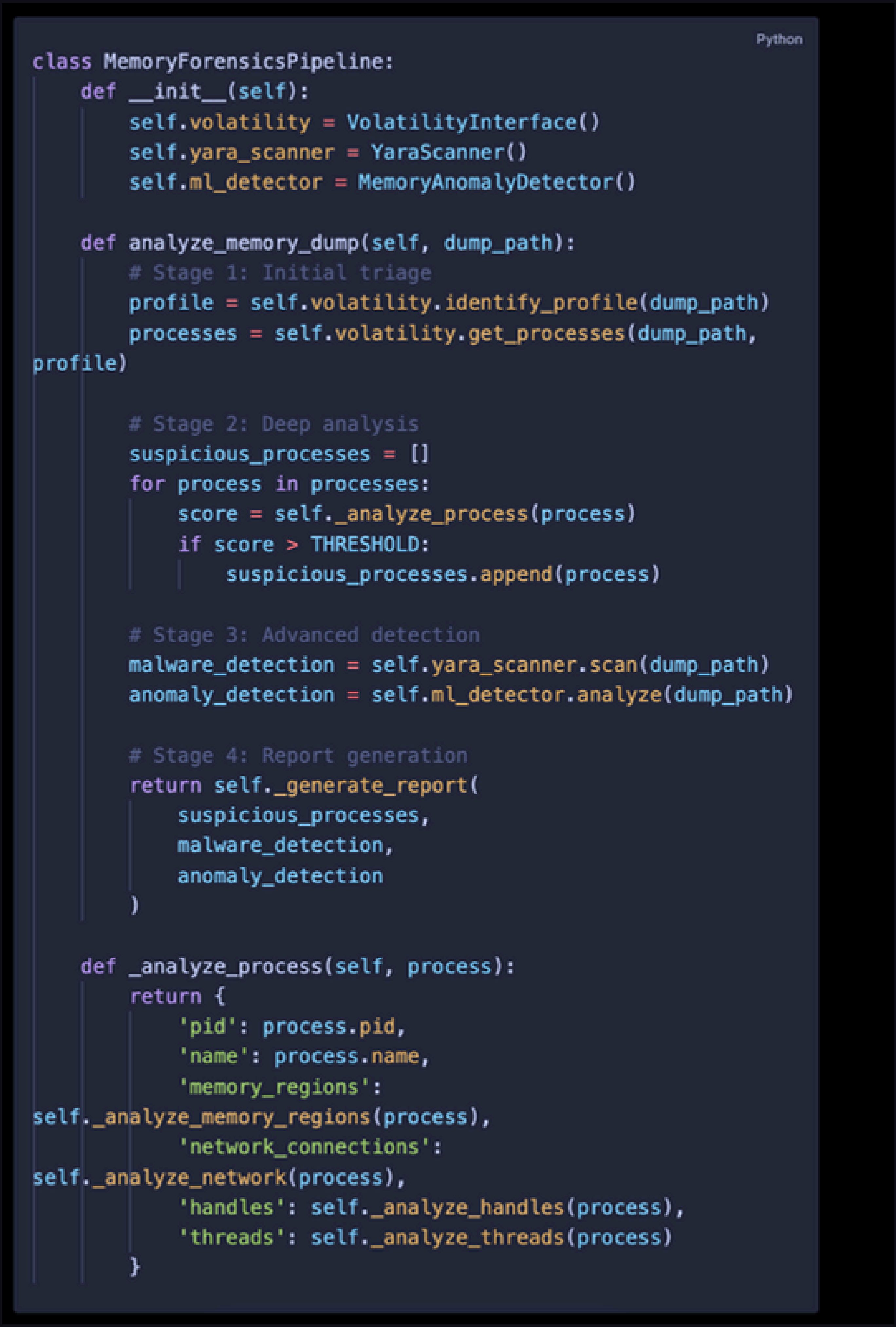


Automated Analysis Pipeline

- Implement automated triage systems for memory dumps
- Develop intelligent filtering mechanisms for relevant artifacts
- Create automated reporting systems

Continuous Monitoring

- Implement real-time memory monitoring systems
- Develop automated alert mechanisms for suspicious memory activities
- Create systems for continuous baseline comparison



```
class MemoryForensicsPipeline:
    def __init__(self):
        self.volatility = VolatilityInterface()
        self.yara_scanner = YaraScanner()
        self.ml_detector = MemoryAnomalyDetector()

    def analyze_memory_dump(self, dump_path):
        # Stage 1: Initial triage
        profile = self.volatility.identify_profile(dump_path)
        processes = self.volatility.get_processes(dump_path,
profile)

        # Stage 2: Deep analysis
        suspicious_processes = []
        for process in processes:
            score = self._analyze_process(process)
            if score > THRESHOLD:
                suspicious_processes.append(process)

        # Stage 3: Advanced detection
        malware_detection = self.yara_scanner.scan(dump_path)
        anomaly_detection = self.ml_detector.analyze(dump_path)

        # Stage 4: Report generation
        return self._generate_report(
            suspicious_processes,
            malware_detection,
            anomaly_detection
        )

    def _analyze_process(self, process):
        return {
            'pid': process.pid,
            'name': process.name,
            'memory_regions':
self._analyze_memory_regions(process),
            'network_connections':
self._analyze_network(process),
            'handles': self._analyze_handles(process),
            'threads': self._analyze_threads(process)
        }
```

Conclusion

In conclusion, memory forensics is a vital pillar of modern digital investigations, offering unique insights into the volatile data that underpins system activity. By capturing and analyzing memory, investigators can uncover critical evidence of advanced threats, including malware, unauthorized access, and system misconfigurations that evade traditional forensic methods. This guide has highlighted the essential tools, techniques, and best practices necessary to excel in this field, underscoring its importance in incident response, malware analysis, and proactive threat hunting. As cyber threats evolve, mastering memory forensics equips professionals with the expertise needed to detect, analyze, and mitigate even the most sophisticated attacks, ensuring robust system security and resilience.

HADESS

cat ~/.hadess

"Hadess" is a cybersecurity company focused on safeguarding digital assets and creating a secure digital ecosystem. Our mission involves punishing hackers and fortifying clients' defenses through innovation and expert cybersecurity services.

Website:

WWW.HADESS.IO

Email

MARKETING@HADESS.IO

To be the vanguard of cybersecurity, Hadess envisions a world where digital assets are safeguarded from malicious actors. We strive to create a secure digital ecosystem, where businesses and individuals can thrive with confidence, knowing that their data is protected. Through relentless innovation and unwavering dedication, we aim to establish Hadess as a symbol of trust, resilience, and retribution in the fight against cyber threats.